Stock Picking via Nonsymmetrically Pruned Binary Decision Trees With Reject Option

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

doctor rerum politicarum
(Doktor der Wirtschaftswissenschaft)
im Fach Statistik

eingereicht an der
Wirtschaftwissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
M.Sc. Anton V. Andriyashin
geboren am 10.11.1981 in Rejkjavik, Island

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Christoph Markschies

Dekan der Wirtschaftwissenschaftlichen Fakultät:
Prof. Oliver Günther, Ph.D.

Gutachter:
1. Prof. Dr. Wolfgang Härdle
2. P.D. Dr. Marlene Müller

eingereicht am: 21. Oktober 2008
Tag der mündlichen Prüfung: 5. November 2009

Abstract

Stock picking is the field of financial analysis that is of particular interest for many professional investors and researchers. There is a lot of research evidence supporting the fact that stock returns can effectively be forecasted. While various modeling techniques could be employed for stock price prediction, a critical analysis of popular methods including general equilibrium and asset pricing models; parametric, non- and semiparametric regression models; and popular black box classification approaches is provided. Due to advantageous properties of binary classification trees including excellent level of interpretability of decision rules, the trading algorithm core is built employing this modern nonparametric method. Optimal tree size is believed to be the crucial factor of forecasting performance of classification trees. While there exists a set of widely adopted alternative tree induction and pruning techniques, which are critically examined in the study, one of the main contributions of this work is a novel methodology of nonsymmetrical tree pruning with reject option called Best Node Selection (BNS). An important inverse propagation property of BNS is proven that provides an easy way to implement the search for the optimal tree size in practice. Traditional cost-complexity pruning shows similar performance in terms of tree accuracy when assessed against popular alternative techniques, and it is the default pruning method for many applications. BNS is compared with cost-complexity pruning empirically by composing two recursive portfolios out of DAX30 stocks. Performance forecasts for each of the stocks are provided by constructed decision trees that are updated when new market information becomes available. It is shown that BNS clearly outperforms the traditional approach according to the backtesting results and the Diebold-Mariano test for statistical significance of the performance difference between two forecasting methods. Another novel feature of this work is the use of individual decision rules for each stock as opposed to pooling of learning samples, which is done traditionally. Empirical data in the form of provided individual decision rules for a randomly selected time point in the backtesting set justify this approach.

Keywords:
Binary Decision Trees, Stock Picking, Nonsymmetrical Tree Pruning, Reject Option

Zusammenfassung

Die Auswahl von Aktien ist ein Gebiet der Finanzanalyse, die von speziellem Interesse sowohl für viele professionelle Investoren als auch für Wissenschaftler ist. Empirische Untersuchungen belegen, dass Aktienerträge vorhergesagt werden können. Während verschiedene Modellierungstechniken zur Aktienselektion eingesetzt werden könnten, analysiert diese Arbeit die meist verbreiteten Methoden, darunter allgemeine Gleichgewichtsmodelle und Asset Pricing Modelle; parametrische, nichtparametrische und semiparametrische Regressionsmodelle; sowie beliebte Black-Box Klassifikationsmethoden. Aufgrund vorteilhafter Eigenschaften binärer Klassifikationsbäume, wie zum Beispiel einer herausragenden Interpretationsmöglichkeit von Entscheidungsregeln, wird der Kern des Handelsalgorithmus unter Verwendung dieser modernen, nichtparametrischen Methode konstruiert. Die optimale Größe des Baumes wird als der entscheidende Faktor für die Vorhersageperformance von Klassifikationsbäumen angesehen. Während eine Vielfalt alternativer populärer Bauminduktions- und Pruningtechniken existiert, die in dieser Studie kritisch gewürdigt werden, besteht eines der Hauptanliegen dieser Arbeit in einer neuartigen Methode asymmetrischen Baumprunings mit Abweisungsoption. Diese Methode wird als Best Node Selection (BNS) bezeichnet. Eine wichtige inverse Fortpflanzungseigenschaft der BNS wird bewiesen. Diese eröffnet eine einfache Möglichkeit, um die Suche der optimalen Baumgröße in der Praxis zu implementieren. Das traditionelle cost-complexity Pruning zeigt eine ähnliche Performance hinsichtlich der Baumgenauigkeit verglichen mit beliebten alternativen Techniken, und es stellt die Standard Pruningmethode für viele Anwendungen dar. Die BNS wird mit cost-complexity Pruning empirisch verglichen, indem zwei rekursive Portfolios aus DAX-Aktien zusammengestellt werden. Vorhersagen über die Performance für jede einzelne Aktie werden von Entscheidungsbäumen gemacht, die aktualisiert werden, sobald neue Marktinformationen erhältlich sind. Es wird gezeigt, dass die BNS der traditionellen Methode deutlich überlegen ist, und zwar sowohl gemäß den Backtesting Ergebnissen als auch nach dem Diebold-Marianto Test für statistische Signifikanz des Performanceunterschieds zwischen zwei Vorhersagemethoden. Ein weiteres neuartiges Charakteristikum dieser Arbeit liegt in der Verwendung individueller Entscheidungsregeln für jede einzelne Aktie im Unterschied zum traditionellen Zusammenfassen lernender Muster. Empirische Daten in Form individueller Entscheidungsregeln für einen zufällig ausgesuchten Zeitpunkt in der Überprüfungsreihe rechtfertigen diese Methode.

Schlagwörter:
Binäre Entscheidungsbäume, Aktienselektion, Asymmetrisches Baumpruning, Abweisungsoption

Acknowledgement

This work, in its present form, would not be possible to conduct without the kind support of *DekaBank* and its PhD scholarship program. I hope the novel elements of this study and obtained empirical results will be beneficial for asset managers of the company.

I would like to express my sincere gratitude to Prof. Yuri Nikolaevich Cheremnykh, Prof. Igor Germogenovich Pospelov, and Prof. Emil Borisovich Ershov for introducing me to the world of economic modeling and quantitative analysis. Their valuable advises, support, and constant challenge of reaching new scientific heights are greatly appreciated.

I would like to thank Prof. Wolfgang Härdle for making me familiar with the topic of classification and regression trees.

It is a pleasure to thank Oliver Blaskowitz and other fellow PhD students for their hints, some of which were extremely helpful.

I am grateful for a suggestion of David Philpotts from *Schroder Investment Management*.

I would also like to express my appreciation to Andreas Merk who helped me with the German translation and gave several valuable advises.

And certainly my special thanks goes to all members of the Institute for Statistics. Our joint research activities have always been great, and I hope it stays that way in future.

Berlin, October 2008

Anton V. Andriyashin

Non scholae, sed vitae discimus

Contents

List of Figures

List of Tables

1 Introduction

Professional capital management involves numerous forms of asset allocation and employment of various financial instruments. Trying to obtain better risk-return characteristics, available funds are frequently invested into different stocks constituting a diversified portfolio. The components of such a portfolio are to be regularly revised, and at this point individual stock performance is what counts.

There is a lot of research evidence supporting the fact that stock returns can effectively be forecasted. Fama and French (1988b) conclude that in tests for the 1926-1985 period (New York Stock Exchange – NYSE – 1-month returns), large negative autocorrelations for return horizons beyond a year suggest that predictable price variation due to mean reversion accounts for large fractions of 3-5-year return variances. Predictable variation is estimated to be about 40% of 3-5-year return variances for portfolios of small firms, and the percentage falls to around 25% for portfolios of large firms. In Keim and Stambaugh (1986) it is concluded that several predetermined variables that reflect levels of bond and stock prices appear to predict returns on common stocks of firms of various sizes, long-term bonds of various default risks, and default-free bonds of various maturities.

Moreover, as in Fama and French (1988a) and Balvers et al. (1990), it is argued that predictability is not necessary inconsistent with the concept of market efficiency. In Fama and French (1988a) dividend yields, which are dividend-price ratios, were used to forecast returns on the value- and equally weighted portfolios of NYSE stocks for return horizons (holding periods) from one month to four years. It is shown that the predictable (expected) component of returns is a small fraction of short-horizon return variances, and the power of dividend yields to forecast stock returns, measured by regression R^2, increases with the return horizon. Fama (1991) examines the links between expected returns and macro-variables and acknowledges the existence of connection between expected returns and shocks to tastes or technology (changes of business conditions). Chen (1991) continues the work in this direction and concludes the consistency of the link between excess return macro-variables and growth rates of output with intertemporal asset-pricing models. Balvers et al. (1990) examine the intertemporal general equilibrium model that is the standard neoclassical growth model with serial correlation in aggregate output in order to relate financial asset returns to movements in aggregate output. It is concluded that stock returns in this model can be predicted based on rational forecasts of output, while the empirical results confirmed theoretical implications of the model.

Hodrick (1992) explores alternative ways of conducting inference and measurement for long-horizon forecasting with an application to dividend yields as predictors of stock returns. The application investigates the predictability of stock returns at five horizons, from one month to four years, and the VAR (vector autoregression) tests provide strong evidence of the predictive power of one-month-ahead returns at least for the sample from 1952 to 1987 (Center for Research in Security Prices data). The estimates and Monte-

Carlo results support the conclusion that changes in dividend yields forecast significant persistent changes in expected stock returns.

Shiller (1990) argues that speculative asset prices tend to show excess volatility relative to simple present value efficient market models, and that prices are partly forecastable as tending to returning to 'mean', appropriately defined. While the aforementioned works tend to explain the variation in stock returns attributing it to responses in macroeconomic factors, Shiller (1989) judges that irrational swings in investor sentiment are the prime moving force (Fama and French, 2002).

Jegadeesh (1990) presents new empirical evidence of predictability of individual stock returns. The results documented in this paper reliably reject the hypothesis that the stock prices follow random walks. The author concludes that predictability of stock returns can be attributed either to market inefficiency or to systematic changes in expected stock returns. It is pointed out that empirical results appear quite striking – it follows that the extent to which security returns can be predicted based on past returns is economically significant.

Conrad and Kaul (1998) employ a single unifying framework to analyze the sources of profits to a wide spectrum of return-based trading strategies implemented in the literature and showed that less than 50% of the 120 strategies implemented in the article yield statistically significant profits.

Lewellen and Shanken (2002) acknowledge the evidence that stock returns are predictable and focus on the interpretation of predictability. While in the common framework consistent with the notion of market efficiency (Fama, 1970) researchers must judge whether predictability is consistent with rational behavior or whether it is better explained by irrational mispricing, the authors argue that there is a third potential source of predictability – parameter uncertainty that drives a wedge between the distribution perceived by investors and distribution estimated by empirical tests. The authors also agree with the observations of Stulz (1987) and Lewis (1989) when they point out that prices can appear to react inefficiently to information simply because investors learn about the economy. It is concluded that learning can significantly affect asset-pricing tests, and predictability is not due to some spurious estimation problem, but, rather, it is a feature of the true data-generating process, which corresponds to conclusions of Fama (1998b) who argues that various long-horizon return anomalies in the literature are chance results, consistent with market efficiency – apparent overreaction to information is about as common as underreaction, and post-event continuation of pre-event abnormal returns is about as frequent as post-event reversal.

Given the amount of supporting evidence on stock returns predictability, this applied work, partly motivated by the valuable collaboration with one top financial services company, focuses on the practical ability to forecast stock returns effectively rather than on theoretical explanations of this phenomenon. Using the available market data, next period stock price movements are predicted using an advanced technique of modern nonparametric multivariate analysis called *binary decision trees*. Binary decision trees are a classification method that was introduced in 1980s by a group of American scientists and is thoroughly described in Breiman et al. (1987).

Many studies like Ferson and Harvey (1991) or Campbell and Hamao (1992) employ standard statistical and econometric methods to examine predictability of excess stock returns. However, special properties of decision trees create a notorious distinction among the pool of other available classification techniques. Unlike parametric methods,

which are quite sensitive to issues of misspecification, one of the advantages of decision trees (or Classification and Regression Trees – CART – as they are called alternatively) is the ability to handle specification issues much smoother. Moreover, the nature of the method provides substantial benefits for the classification result interpretation, see Breiman et al. (1987) for more details. Steadman et al. (2000) emphasize the practical importance and flexibility of decision trees in the way that this method poses contingent – and thus possibly different – questions to classify an object into a given set of classes, while the traditional parametric regression approach employs the common set of questions for each classified object, and final classification score is produced by weighting every answer. Moreover, a parametric regression relies on a particular error distribution assumption (e.g. the Gaussian model), and decision trees become particularly useful when the data do not meet this assumption (Feldman et al., 1972).

In the recent years several financial services companies (e.g. *JPMorgan* and *Salomon Smith Barney*) showed their interest in applying decision trees for stock picking by issuing a number of press releases for professional investors (Brennan et al., 1999; Seshadri, 2003; Sorensen et al., 1999). The reports provide valuable feedback on the method performance potential when decision trees are applied to the US stock market. This study extends the geography of the method application and focuses on the German DAX30 stock market.

Decision tree financial applications are not limited solely by the stock selection challenge. In Schroders (2006) the selection of underperforming and outperforming Pan-European banks is achieved with the help of decision trees, and asset allocation to shares, bonds, or cash is also derived with the help of CART in Harman et al. (2000).

Apart from purely economic settings, decision trees are successfully employed in particle identification in nuclear physics (Roe et al., 2005), prediction of daily maximum surface ozone concentration in meteorology (Burrows et al., 1995), phrase break prediction in speech synthesis (Kim and Oh, 2007), lung cancer detection (Härdle et al., 2007), genetic programming (Koza, 2007), and many other applications.

Many studies employing CART use the industry-standard approach of tree building described in Breiman et al. (1987). Some studies prefer to use other popular tree building techniques such as C4.5, however various empirical comparisons suggest that produced accuracy levels are very close, see Section 6.1 for more details. Prior simulations (Kim and Loh, 2001) and architecture of the classical method have shown that due to the specific nature of financial markets, it might be reasonable to change the classical approach and introduce potentially a more effective technique of tree building.

Tree pruning is considered to be the most important step (Breiman et al., 1987) in obtaining a proper decision tree, which potentially can have various sizes. Overfitting or underfitting directly affect, and affect negatively, the forecasting power of such a decision rule. In Schroders (2006) it is mentioned that the traditional tree pruning approach (Breiman et al., 1987) used by the authors in the past is now substituted with a set of three rules based on different decision tree characteristics. Although the algorithms are not revealed explicitly, this statement creates additional motivation to search for a more effective decision tree pruning technique for financial applications.

The main contribution of this work is the presentation of the novel methodology of *nonsymmetrical decision tree pruning with reject option* called *Best Node Selection* (BNS). While the traditional cost-complexity approach operates only with node triplets when pruning (see Chapter 5), BNS allows for more flexible tree optimization and focuses

on individual node characteristics rather than an integral measure of quality of a given subtree.

The efficiency of the new method is examined on DAX30 stock market via backtesting of the stock picking algorithm employing available DAX30 company data for the period of 2002–2004. One important theoretical property of BNS is proven, and backtesting results are compared with the similar trading strategy that relies on the canonical version of tree pruning described in Breiman et al. (1987). According to the Diebold-Mariano test (Diebold and Mariano, 1995), the economic performance difference between the two forecasting methods proved to be significant at the 0.1% confidence level in favor of the novel methodology.

BNS allows not only to achieve significant *economic* benefits but also maintains the very high level of interpretability of produced decision rules and requires little computation time. Unlike some of ensemble methods such as bagging or boosting that have the potential to produce a lower misclassification rate but only at the cost of switching to black box models (see Section 6.3), every BNS decision rule is transparent in terms of employed criteria, significance of variables, and reliability of various parts of the rule. Moreover, BNS automatically suggests when a classification is more likely to be wrong and introduces reject areas where a classification is halted. The end-user can override these decisions of the system if necessary, and risk parameters can be adjusted following the individual needs of a particular financial application. Unlike BNS, other widely used tree-based classification methods do not have reject options, and classifications are performed unconditionally of risk levels of various parts of the rule.

Throughout this work, various modeling techniques and practical challenges are examined. Although some of the methods like boosting are not technically feasible due to the extremely high amount of necessary computations, in industrial settings and for some algorithmic trading applications these black box approaches may be of great interest, and therefore they are critically examined in the study, too.

2 Structure of the Work

The study starts with the motivation for stock picking and describes the assumptions that must be fulfilled for a trading strategy to be considered appropriate such as the high level of interpretability of trading rules, etc. Institutional investors are considered as recipients of the strategy (that enables short sales, for instance), however the work is performed in an academic setting that naturally limits the amount of computations that can be conducted for various trading strategies.

Chapter 3 focuses on the critical analysis of various popular modeling techniques for stock pricing and forecasting. Section 3.2 examines general equilibrium and popular asset pricing models including the CAPM, the ICAPM, and the APT. The link between a special version of the consumption-based model, the CAPM, and other pricing models is analyzed. In Section 3.3, parametric regression models are considered as a means to estimate and apply in practice asset pricing models. Distributional assumptions are questioned and properties of estimators in various setups are compared. The issue of the correct specification is found to be one the most important, and because it is impossible to obtain the correct structure endogenously, non- and semiparametric regression models are considered to be more flexible alternatives, which are reviewed in Section 3.4. It is concluded that nonparametric regression models can be used efficiently only in the case when there is one or two explanatory variables due to the so called curse of dimensionality and limited possibilities of graphical interpretation. To cope with these challenges, various dimension reduction techniques are employed resulting in the so called semiparametric methods that combine a nonparametric regression and parametric estimation of a linear index function. A critical overview of the well known models is provided, and it is concluded that although semiparametric models are a major step forward, their flexibility comes at the cost of imposing extra structural assumptions, which can become a serious side-effect that neglects all other advantageous features when misspecification of the model is severe. Semiparametric models do not induce the significant explanatory variables endogenously, therefore the end-user must select input variables correctly, which is problematic in many cases.

Section 3.5 considers statistical alternatives – the layer of classification techniques. The Bayes discriminant rule faces the challenge of correct estimation of a multidimensional variable density, linear discriminant and quadratic discriminant analysis are black box models that can be too rigid for many circumstances, the k-nearest neighbor estimator suffers from the curse of dimensionality when many explanatory variables are employed. Artificial Neural Networks appear to be a powerful method capable of producing nonlinear decision rules, however the method exhibits some severe pitfalls – there is a risk of overfitting in terms of employed hidden layers; initial values of the parameters are quite important for the numerical derivation of the optimal rule, and it is uncommon to run the procedure many times to find the global optimum since the amount of computations would become overwhelming. Finally, Artificial Neural Networks can be extremely difficult – if not impossible – to interpret. Support Vector Machines are a set

of other powerful nonlinear classification and regression techniques that are comparable to Artificial Neural Networks. However, Support Vector Machines can be very sensitive to outliers in the training sample and may require longer computation time. The method does not induce the explanatory variable set automatically, and it is impossible to interpret produced decision rules easily. Classification and regression trees seem to tackle the majority of the aforementioned challenges quite effectively, and the technique is therefore proposed as the core for stock picking.

Chapter 4 formally introduces binary classification trees. It starts with describing the major steps of tree building and application: creation of the maximum tree, tree pruning, and classification of new data with the optimized decision rule. The next sections concentrate on the impurity measure definition and its properties, special forms of the impurity measure, and some practical examples how various measure specifications influence the shape of produced trees.

Chapter 5 focuses on the traditional way of tree pruning via the cost-complexity tradeoff. The relative inefficiency of early-stopping rules and nonmonotonicity of the impurity measure decrement, illustrated in Section 5.1, provide substantial motivation for more complex methods of pruning. Cross-validation is defined in Section 5.2, and its empirical properties are carefully examined. Section 5.3 formally describes the standard way of tree pruning via the cost-complexity criterion and cross-validation of endogenously induced subtrees. An empirical rule of thumb for the final tree size selection is introduced there as well.

Chapter 6 considers alternative tree induction and tree pruning techniques that may potentially be more suitable for the stock picking core. Decision tree induction techniques, analyzed in Section 6.1, include FACT/QUEST, ID3/C4.5, CHAID, oblique and nonlinear decision trees, and random splitting. It is concluded that after standard pruning, the accuracy of produced trees is rather comparable, although the final decision rule structures may be different in terms of their size. Standard axis-parallel trees can be named among the most versatile and efficient choices especially when the decision rule interpretability is an important factor. Alternative pruning techniques, considered in Section 6.2, include critical value pruning, minimum-error pruning, reduced-error pruning, pessimistic error and error-based pruning, Minimum Description Length pruning, and pruning using multiple performance measures. It is concluded that cost-complexity pruning and other methods generally perform comparably with cost-complexity trees, however the latter tend to contain fewer nodes, which is an indicator of higher relative rule interpretability. Multiple performance measures pruning is an attempt to meet potentially more sophisticated needs of the end-user when the accuracy measure does not suffice to describe fully the 'quality' of a decision rule. Many elements in such pruning are user-defined (for instance, utility functions or various threshold values), therefore the direct comparison of this pruning method with the other aforementioned techniques is not feasible. Section 6.3 introduces several well known machine learning ensemble methods and their applications to single decision tree classifiers. Bagging, adaptive boosting, and random forests are carefully examined from the perspective of alternative stock picking applications. Although these methods are pure black box classification approaches and at least two of them are rather slow, their potential application is analyzed in the realm of alternative asset management applications like high-frequency arbitrage trading.

One of the main contributions of this study is the introduction of a novel pruning

technique – Best Node Selection (Chapter 7). Unlike traditional pruning methods, Best Node Selection provides a more flexible approach of tree handling and allows nonsymmetrical pruning when only one of child nodes can be pruned if necessary. Instead of assessing a tree by means of an integral measure, Best Node Selection focuses on individual node characteristics. Section 7.2 proposes two user-defined measures of node reliability that are based on class purity and representativity, which is illustrated by several examples. Best Node Selection combines the nonsymmetrical structure of tree pruning and reject option that is discussed in Section 7.3. Rigorous formulation of the method and its inverse propagation property, which is quite beneficial for practical implementation, are provided in Section 7.4. Section 7.5 illustrates the power of the method when noisy input data are considered.

The aim of next part of the work is to compare empirically the standard and novel pruning techniques by historical simulation of DAX30 stock trading in Chapter 8. Section 8.1 describes the general setup and available data, Section 8.2 analyzes the so called 'big hit ability' (or the ability of a method to forecast effectively the movements with big relative magnitude) and class assignment rules. Two nested competing input variable specifications and two types of the learning sample creation are introduced in Section 8.3, and motivation for their use in practice is discussed. Section 8.4 focuses on the parameter calibration and provides the relevant empirical results. Backtesting and recursive portfolio creation are addressed in Section 8.5. Portfolio returns and risk characteristics are analyzed for competing trading strategies. Best Node Selection and cost-complexity pruning, which are a driving force of relevant portfolios performance, are illustrated on the same underlying tree. Given the similar hit ratios but rather different risk-return portfolio characteristics, the obtained financial results are examined for spuriousness in Section 8.6. According to the Diebold-Mariano test, the economic value associated with portfolio returns generated by the Best Node Selection and cost-complexity decision tree pruning strategies is significantly different in favor of Best Node Selection at any reasonable confidence level.

Chapter 9 briefly summarizes the main results of the study and addresses valuable experience of professional asset managers in a similar setup. Another novel feature of this study – the employment of independent decision trees for each analyzed stock – is illustrated for a randomly selected time point of the backtesting period. Practical limitations of replicating the introduced investment strategy for individual investors are discussed.

Several statements related to Best Node Selection are formulated and proved in the Appendix. The Appendix also contains performance charts for each of the backtested stocks. To be able to compare individual stock price forecasting rules, decision trees for the same randomly selected time point of the backtesting set are provided in the Appendix, too.

3 Stock Picking Challenge and Modeling Opportunities

3.1 The Challenge of Stock Picking

Market data contain useful information about underlying companies, and this information has at least partial but quite significant impact on future stock prices as it can be seen from the rich scientific evidence overviewed in Chapter 1.

Let P_t be the current stock price of some company and matrix $\boldsymbol{X}_t$ represent the accumulated available market data up to the current moment (for instance, fundamental and technical data from one of the major financial data providers). If a certain link between $\boldsymbol{X}_t$ and P_{t+1}, i.e. the next period (and yet unknown) stock price, is assumed, then one of the most important questions is the way this link is to be reconstructed.

If the next period stock price is forecasted correctly, then the long (if the stock price is expected to rise) or short (if the stock price is expected to fall) position opened at date t will yield

$$|R_t| = \left| \frac{P_{t+1} - P_t}{P_t} \right| \tag{3.1}$$

after one period, where R_t is the one-period-ahead stock return. Note that the value of R_t is subject to transactions costs when the actual profit of a trading strategy is to be calculated.

The ultimate goal of this study is to present a working methodology of effective stock picking that comes as a result of the critical analysis of strengths and weaknesses of various techniques from economics, statistics, and econometrics, implement this methodology, and practically test it on the German stock market (DAX30) data. The overwhelming empirical evidence, presented in Chapter 1, provides reasons to believe that even on efficient stock markets there are certain links between changes in stock prices and technical and fundamental underlying data.

Practical experience, consultations with investment analysts, and technical constraints (academic setting) suggest that produced stock performance classification rules should meet certain important requirements:

- each classification rule should be highly interpretable so that the end-user carrying out the final investment decision has the significant amount of information for the analysis;
- various factors entering the decision rule should be easily identifiable in terms of their relative significance in the classification;
- means to identify classification risks as well as the end-user control over built-in (if any) risk measures are highly desirable but not essential;

- the amount of required computations should be reasonable since no industrial setting is available to conduct this work at the moment.

As it was pointed out in Chapter 1, some financial applications may not require the high level of interpretability of produced classification rules – this could be the case, for instance, in high frequency arbitrage trading. Many other investment problems, however, require sound reasons for any particular important decision, and this work is focused primarily on the second setting. However, relevant and competitive black box techniques are not automatically neglected on the simple basis of being difficult or impossible to interpret, and they are carefully examined for alternative practical applications, too.

It is assumed that modeled trading strategies are aimed at institutional investors, which implies the direct possibility of short sales.

The next sections provide a critical overview of the most popular methodologies that could be employed for stock price modeling and motivate the choice of decision trees (a modern nonparametric technique) as the stock picking core that is introduced in this work.

3.2 Traditional Asset Pricing

Traditional asset pricing focuses on modeling of global economic systems where an asset price (or prices of different assets) is a particular object of interest. Following Cochrane (2005), let us consider here the simplest possible case for illustrative purposes where behavior of some investor is modeled.

Let c_t be the current consumption, P_t – stock price at time t, D_{t+1} – dividends paid at time t, $X_{t+1} = P_{t+1} + D_{t+1}$ – next period payoff (unknown to an investor). If β is the subjective discount factor, then the utility function, defined over current and future values of consumption, is assumed to take the following form:

$$U(c_t, c_{t+1}) = u(c_t) + \beta \mathbb{E}_t \left[u(c_{t+1}) \right] \tag{3.2}$$

where $\mathbb{E}_t$ is the conditional (on the present date t) expected value operator.

If e is the original consumption level (if the investor bought none of the asset) and given the unconstrained possibility to buy or sell as much of the asset at a price P_t, the optimization problem is:

$$\begin{cases} \max\limits_{\xi} u(c_t) + \mathbb{E}_t[\beta u(c_{t+1})] \; s.t. \\ c_t = e_t - P_t \xi, \\ c_{t+1} = e_{t+1} + x_{t+1} \xi, \\ X_{t+1} = P_{t+1} + D_{t+1} \end{cases} \tag{3.3}$$

where ξ is the amount of the asset he chooses to buy.

The first-order condition for the optimal consumption and portfolio choice is equivalent to

$$P_t = \mathbb{E}_t \left[\beta \frac{u'(c_{t+1})}{u'(c_t)} x_{t+1} \right] \tag{3.4}$$

or

$$P_t = \mathbb{E}_t \left(m_{t+1} x_{t+1} \right) \tag{3.5}$$

where

$$m_{t+1} = \beta \frac{u'(c_{t+1})}{u'(c_t)} \tag{3.6}$$

is the *stochastic discount factor* or *pricing kernel* as it is called sometimes.

Although (3.3) represents one of the simplest models, its solution – the equation (3.4) – outlines the further steps that are necessary to be undertaken when the precise solution for one or more assets (in a possibly more complicated model) should be found. Given the payoff x_{t+1} and endogenous investor's consumption choice c_t and c_{t+1}, the equation (3.4) computes the expected market price.

Sometimes results similar to (3.4) are criticized for lacking practical applicability – to get the optimal trajectory for the asset price, one has to assume a specific form of the utility function and somehow model the future payoff flow, which takes the form of either an exogenous stochastic process or optimal trajectory from some other dynamic optimization task when the general equilibrium setting is assumed.

One should not, however, be too skeptical about evidence that can be obtained from such type of modeling. Major market movements as well as some investor incentives can be analyzed using equations like (3.4) when *general intertemporal equilibrium* models are built. Even in a fully deterministic model of the general economic equilibrium with multiple agents and assets, it is still possible to obtain nontrivial transition patterns of the optimal trajectories. When optimization tasks are solved for each of the agents and all balancing and transversality conditions (Pontryagin et al., 1962) are properly accounted, asset prices can be obtained as functions of other better measured macroeconomic aggregates. For examples of such models and thorough discussion on their motivation, derivation, and examples of practical applications refer, for instance, to Petrov et al. (1996) and Pospelov (2003).

Another well-known critique about the equation (3.4) is that in reality there are no representative agents, investors do not optimize some virtual utility function, etc., and therefore a much better way to predict asset prices is to use one of the factor pricing models such as the CAPM (Sharpe, 1964; Lintner, 1965b,a), the ICAPM (Merton, 1973), or the APT (Ross, 1976) that seem to avoid the use of such notions as a utility function at all. However, this statement is not true for the following reasons.

First of all, the equation (3.4) does not assume the completeness of markets or existence of a representative investor. No special form of the utility function is assumed at that step as well. Payoffs must not follow some specific distribution and must not be independent over time. Finally, no assumption about the market equilibrium is made at this step, too.

It is true, however, that the equation (3.4) can not be solved without making some of the aforementioned assumptions. But it is also true that the CAPM and the ICAPM are general equilibrium models with linear technologies, i.e. just *a special case of the consumption-based model* where a discount factor is specified as a linear function of a set of proxies. The CAPM can be derived (Cochrane, 2005) from the consumption-based model by:

- two-period quadratic utility;

- two periods, exponential utility, and normal returns;
- infinite planning horizon, quadratic utility, and *i.i.d.* (independent and identically distributed) returns;
- log utility $u(c) = \ln(c)$.

As a result of the derivation, the CAPM in fact specifies the pricing kernel as a linear function of the following form:

$$m_{t+1} = a + bR^W_{t+1} \tag{3.7}$$

where R^W is the 'wealth portfolio', which conventional proxies are returns on a broad-based stock portfolio such as NYSE, S&P500, etc., and a, b are parameters.

The equation (3.7) is more frequently stated in the equivalent return-beta form:

$$\mathbb{E}(R^i) = \gamma + \beta_{i,R^W}\left[\mathbb{E}(R^W) - \gamma\right] \tag{3.8}$$

where

$$\beta_{i,R^W} = \frac{cov(R^W, R^i)}{var(R^W)} \tag{3.9}$$

is the exposure to the market risk.

There is, however, evidence that the sensitivity of expected stock's return to the market return – *beta* – does not suffice to describe the expected return, see Fama and French (1996) for more details.

The ICAPM expands the CAPM by transferring from the single risk factor (market) to multiple risk factors that are growth rates of state variables including aggregate wealth:

$$m_{t+1} = a + b^\top f_{t+1} = a + b_A f^A_{t+1} + b_B f^B_{t+1} + \dots \tag{3.10}$$

where f^i are factors and a, b_i are parameters.

The main challenge with the ICAPM is that the set of variables is too broad. Fama (1998a) raises the question of identifying these risk factors and concludes that ignoring estimation problems, it is theoretically possible to find the set of priced state variables (which give rise to special risk premiums in expected returns) when the state variables are named. When the state variables are not named but their total number is known, even the number of them that produce special risk premiums is probably impossible to determine. In principle, any variable that forecasts future returns can enter the model, but it is not known exactly how many factors are there that affect stock returns.

Although the APT relies on different assumptions and, unlike the CAPM and ICAPM, does not require the economic structure, in practice the difference between the application of the APT and ICAPM is not significant.

Formally, the APT states that if a set of asset returns is generated by a linear factor model

$$R^i = \mathbb{E}(R^i) + \sum_{j=1}^{N} \beta_{ij}\tilde{f}_j + \varepsilon^i, \quad \mathbb{E}(\varepsilon^i) = \mathbb{E}(\varepsilon^i \tilde{f}_j) = 0, \tag{3.11}$$

then with additional assumptions there is a discount factor $m = a + b^\top f$ that prices the returns.

Cho et al. (1986) test the APT in an international setting and finally reject the joint hypothesis that the APT is valid internationally. This rejection, however, can not be clearly attributed either to the segmentation of capital markets or failure of the international APT.

The APT uses returns on broad-based portfolios derived from a factor analysis of the return covariance matrix so that portfolios characterizing a common movement are found. In the ICAPM there is no assumption that factors f in the pricing model $m = b^\top f$ describe the covariance matrix of returns, and factors can be specified choosing from the state variables that describe the conditional distribution of future asset returns. But ultimately, practical applications end up with testing various models $m = b^\top f$.

3.3 Parametric Regression Models

The aforementioned evidence suggests that there are successful attempts to find some common factors that drive stock returns, but these factors may be not stable over various assets, regions, and time periods. While that may be a particular problem for rigorous asset pricing modeling, practical applications of asset pricing are more result-oriented and model performance for a given asset is what counts.

The standard econometric approach may loosen some of the structural requirements like that factors in a pricing model must be state variables or describe the covariance matrix of returns. The ICAPM and APT milestones could be taken as the initial hint for model specification that later could be adjusted more flexibly.

If one considers the standard econometric analysis, then the free choice of input variables and potential ability to incorporate structural changes, i.e. when the regression model coefficients vary over time, are among the key benefits. At the same time, the classical parametric regression of an arbitrary specification still contains implied structure limitations and is quite sensitive to misspecification.

Suppose that $\boldsymbol{X}_t^* \subseteq \boldsymbol{X}_t$ is the subset of variables that forms the true data-generating process (DGP) for the next period stock return R_{t+1} at time t. Given the form of the APT as in the equation (3.11), it would be natural to estimate the following linear regression model:

$$Y = \boldsymbol{X}^* \beta + u \tag{3.12}$$

where $\boldsymbol{X}^* \equiv \boldsymbol{X}_t^*$, Y is the vector of observations for next period yields R_{t+1} for a given stock, and

$$u \sim i.i.d. \left(\mathbf{0}, \sigma^2 \boldsymbol{I}\right) \tag{3.13}$$

is the disturbance.

Although the variables from $\boldsymbol{X}^*$ do not necessarily have to be growth rates of state variables (as in the ICAPM) or factors from the covariance matrix of returns (as in the APT), the Gauss-Markov theorem – refer, for instance, to Davidson and MacKinnon (2004), – which guarantees the best (in terms of variance) linear unbiased OLS estimator of β in (3.12), imposes the following constraints:

$$\begin{cases} \mathbb{E}\left\{u | \boldsymbol{X}^*\right\} = \mathbf{0}, \\ \mathbb{E}\left\{uu^\top | \boldsymbol{X}^*\right\} = \sigma^2 \boldsymbol{I} \end{cases} \tag{3.14}$$

that are alternatively called the orthogonality and sphericality assumptions.

Failure to comply with the first, orthogonality, assumption leads to biased estimates whereas failure of the second assumption leads to loss of efficiency although the central tendency of the estimator is still correct (Hausman, 1978).

If two or more predictor variables are highly correlated, which is usually referred to as *multicollinearity*, then difficulties associated with multicollinearity closely resemble the setup when the sample size is too small – in both cases there is not enough information to obtain precise estimates of all the coefficients (Davidson and MacKinnon, 2004).

It is not uncommon when the disturbance variance is not constant across observations, i.e. the regression is *heteroscedastic* (Greene, 1997):

$$\mathbb{V}\left\{u_i|\boldsymbol{x}_i^*\right\} = \sigma_i^2, \quad i = 1, \ldots, n \tag{3.15}$$

where $\boldsymbol{x}_i^*$ is an element of $\boldsymbol{X}^*$, and n is the number of observations in the sample.

At this point, even if the disturbances u_i are assumed to be pairwise uncorrelated, the OLS estimator is less efficient than the GLS one. It is, however, rarely possible to be certain about the nature of heteroscedasticity in a regression model. The weighted least squares estimator

$$\hat{\beta} = \left[\sum_{i=1}^{n} w_i \boldsymbol{x}_i^* (\boldsymbol{x}_i^*)^\top\right]^{-1} \left[\sum_{i=1}^{n} w_i \boldsymbol{x}_i^* y_i\right] \tag{3.16}$$

where the scaling comes from:

$$\sigma_i^2 = \sigma^2 \omega_i, \quad \sum_{i=1}^{n} \omega_i = n \tag{3.17}$$

is consistent regardless of the weights used, as long as the weights are uncorrelated with the disturbances. However, using the wrong set of weights leads to an inefficient weighted least squares estimator.

Heteroscedasticity is most commonly associated with cross-section data. In a time-series setting, the more common problem is *autocorrelation*, or serial correlation of the disturbances across the periods. When historical stock yields from the sample of available data are considered to be the dependent variable, it is the time-series nature of data that emphasizes the challenge of autocorrelation.

For homoscedastic, but correlated across observations, disturbances

$$\mathbb{E}\left\{uu^\top\right\} = \sigma^2 \boldsymbol{\Omega} \tag{3.18}$$

where $\sigma^2\boldsymbol{\Omega}$ is a full, positive definite matrix with a constant $\sigma^2 = \mathbb{V}[u_t]$ on the diagonal, and under the stationarity assumption, the least squares estimate will be unbiased, consistent, and asymptotically normal (under some technical conditions that are met for simple models of autocorrelation like AR(1) and others), but inefficient.

But if the regression contains any lagged values of the dependent variable, the least squares estimate will no longer be unbiased or consistent. And that is quite important because some financial modeling approaches as technical analysis clearly suggest to include lagged stock yields as parts of various indexes and oscillators when obtaining

stock price movement forecasts, see Neftci (1991) and Greene (1997) for more details.

When the linear model (3.12) does not suffice to describe the assumed link between the stock yield and other dependent variables, a *nonlinear* regression can be one of the alternatives:

$$y_t = \boldsymbol{x}_t^*(\beta) + u_t \tag{3.19}$$

where the scalar function $\boldsymbol{x}_t^*(\beta)$ is a nonlinear regression function, and $u_t \sim i.i.d.\,(0, \sigma^2)$ are model residuals as before.

It can be shown that, under the assumption that the error terms are *i.i.d.*, the most efficient MM estimator is nonlinear least squares, see Davidson and MacKinnon (2004) for more details.

In this way – applying MM – one can also directly estimate models described, for instance, by the equation (3.5). The asset pricing model can be then tested applying moment conditions:

$$\mathbb{E}\left[m_{t+1}(b)x_{t+1} - P_t\right] = 0, \tag{3.20}$$

where $b = [\beta \quad \gamma]$ if, for example, $m_{t+1} = \beta(c_{t+1}/c_t)^{-\gamma}$, refer to Cochrane (2005) for more details.

Even more importantly, the models (3.12), (3.19), or any other tested one are assumed to be *correctly specified*, i.e. the DGP that actually generated data belongs to the model under study. Estimating a misspecified regression generally yields biased and inconsistent parameter estimates (Davidson and MacKinnon, 2004). This is also true whenever one or more regressors that are correlated with the regressor included in the model are incorrectly omitted. This suggests that any regression model forecasts are not reliable unless model specification is thoroughly tested.

There are several specification tests that might be employed in this situation (Davidson and MacKinnon, 2004). The first group includes parametric specification tests for linear and nonlinear models such as tests based on artificial regressions and nonnested hypothesis tests. *Akaike information criterion* and *Bayesian information criterion* provide the ways to select the best model (although possibly still not a correctly specified one) out of several competing ones. Finally, there are a number of nonparametric tests that allow to assess the specification of a given parametric model – refer, for instance, to Hong and White (1995) and Johnson and McClelland (1997). Some nonparametric tests are also capable of checking for the existence of heteroscedasticity and autocorrelation when the model is assumed to be specified correctly (Robinson, 1986).

However, if a test fails to confirm the present specification of a regression model, the correct specification does not become automatically available, and there is no guarantee it can ever be found given the available information from the sample, hence possibly compromising the whole parametric regression approach applied to the studied problem of stock picking.

Finally, the estimated classical parametric regression provides the forecasts that are a weighted sum of various independent factors, and these weights (estimated coefficients) do not change from one observation to another, therefore making the model less flexible in explaining various data patterns. Theoretically it is possible to apply the *Chow's breakpoint test* (Chow, 1960) to a given regression model, but the test will just identify if the model is valid for two subsamples. And if it is not valid, then only the existence of this breakpoint becomes known, and not the way how the model should be changed to incorporate structural changes. If several breakpoints are identified, that may ultimately

suggest a severe case of misspecification, but again – not the way how to remedy it.

These serious concerns strongly suggest to consider non- and semiparametric regressions as more flexible and versatile counterparts of traditional parametric regression modeling since no reliable prior knowledge about the functional form of the stock price DGP is available because the aforementioned empirical studies may at best provide only some hints about which input variables are likely to form this DGP.

3.4 Non- and Semiparametric Regression Models

While the parametric regression fits the data to a prespecified linear or nonlinear model, if specification tests reject the current functional form, then generally there is no answer how to find a more suitable parametric form. The nonparametric regression estimates $\mathbb{E}(y_t|x_t)$ directly, without making any assumptions about the functional form. The simplest approach to nonparametric regression is kernel regression (Davidson and MacKinnon, 2004). Two random variables X and Y (both are one-dimensional for the moment) are supposed to be jointly distributed with the probability density function $f(x, y)$, and the goal is to estimate the conditional expectation $m(x) \equiv \mathbb{E}(Y|X = x)$ as a function of x using a sample of observations (y_t, x_t) for $t = 1, \ldots, n$.

When the probability density function of X is unknown, i.e. in the *random design*, see Härdle et al. (2004) for more details, the estimation starts directly with the definition of the conditional expectation:

$$m(x) = \frac{\int y f(x, y) dy}{f_X(x)}. \tag{3.21}$$

The next step is to replace $f(x, y)$ and $f_X(x)$ with their kernel estimates. Estimation of $f_X(x)$ is straightforward and when employing the multiplicative kernel density estimator with the product kernel (Härdle et al., 2004), this leads to the *Nadaraya-Watson estimator*:

$$\hat{m}_h(x) = \frac{n^{-1} \sum_{i=1}^{n} K_h(x - X_i) Y_i}{n^{-1} \sum_{j=1}^{n} K_h(x - X_j)} \tag{3.22}$$

where $K(\cdot)$ is a kernel – a continuous, bounded, and symmetric real function K that integrates to one:

$$\int K(u) du = 1, \tag{3.23}$$

and h is the bandwidth. The shape of the kernel weights is determined by K, whereas the size of the weights is parameterized by h, see Härdle (1990) for more details.

Härdle et al. (2004) conclude that for practical purposes the choice of the kernel function is almost irrelevant for the efficiency of the estimate. As for the bandwidth selection, there is no single best method existing, and even asymptotically optimal criteria may show bad behavior in simulations. As a consequence, it is recommended to determine bandwidths by different selection methods and compare the resulting density estimates. The most commonly used in practice criteria for bandwidth selection are *Silverman's rule of thumb*, *cross-validation*, and *refined plug-in methods*, see Härdle et al. (2004) for an extensive discussion on this and other relevant matters.

The Nadaraya-Watson estimator is the special case of a larger class of kernel regression estimators – it corresponds to a *local constant* least squares fit. *Local linear* and higher order *local polynomial* regressions are suggested by a Taylor expansion of the unknown conditional expectation function $m(\cdot)$:

$$m(t) \approx m(x) + m'(x)(t-x) + \ldots + m^{(p)}(x)(t-x)^p \frac{1}{p!}. \tag{3.24}$$

Local polynomial regression fits a polynomial in a neighborhood of x:

$$\min_{\beta} \sum_{i=1}^{n} \{Y_i - \beta_0 - \beta_1(X_i - x) - \ldots - \beta_p(X_i - x)^p\}^2 K_h(x - X_i) \tag{3.25}$$

where β is the vector of coefficients $(\beta_0, \beta_1, \ldots, \beta_p)^\top$, and weights are controlled by $K_h(x - X_i)$ allowing to vary the neighborhood of x.

The result

$$\hat{\beta}(x) = \left(\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{W} Y \tag{3.26}$$

is a weighted least squares estimator where $\boldsymbol{X}$ is the vector of variables $(X_i - x), \ldots, (X_i - x)^p$, Y is the vector of observations, and $\boldsymbol{W}$ is the weighting diagonal matrix containing $K_h(x - X_i)$ elements. The local polynomial estimator of the regression function m is

$$\hat{m}_{p,h} = \hat{\beta}_0(x) \tag{3.27}$$

where $\hat{\beta}_0(x)$ is the first component of $\hat{\beta}(x) = \left(\hat{\beta}_0(x), \ldots, \hat{\beta}_p(x)\right)$.

For $p = 0$, $\hat{\beta}(x)$ reduces to $\hat{\beta}_0(x)$, which is the local constant estimator, that corresponds to the Nadaraya-Watson estimator. Setting $p = 1$ yields the local linear estimator, which achieves further improvement in the boundary regions. One of the advantages of the local polynomial approach is that it provides an easy way of estimating derivatives of the function $m(\cdot)$. Other popular smoothers include *the nearest-neighbor estimator* (see Section 3.5), *median smoothing*, *spline smoothing*, and others, see Härdle et al. (2004) for more details.

Kernel regression can be generalized for the multidimensional case, i.e. when there are several explanatory variables. Härdle et al. (2004) derive the multivariate Nadaraya-Watson and local linear regression estimators, and analyze their statistical properties. From a practical point of view, though, the main challenge is the so called *curse of dimensionality* – nonparametric regression estimators are based on the idea of local weighted averaging, but in higher dimensions the observations are distributed so sparsely (especially in samples of the limited size) that performance of the respective estimators is not satisfactory. Moreover, a further problem is that for more than two regressors, the graphical interpretation of the results is usually not possible.

To cope with these challenges, various dimension reduction techniques are employed resulting in the so called semiparametric methods that combine a nonparametric regression model and a parametric estimation of a linear index function:

$$\mathbb{E}(Y|\boldsymbol{X}) = m(\boldsymbol{X}) = g\{v_\beta(\boldsymbol{X})\} \tag{3.28}$$

where $g(\cdot)$ is the unknown link function (known link functions such as the Gaussian

cumulative density function lead to well-known regression models like *probit*), and $v_\beta(\cdot)$ is an index function specified up to β, which can take, for instance, the form of $v_\beta = \boldsymbol{X}\beta$ as in classical parametric regression.

Some of the well known semiparametric setups include the following models (Härdle et al., 2004). The *Additive Model* (AM) generalizes the multiple regression model by introducing one-dimensional nonparametric functions:

$$\mathbb{E}\left(Y|\boldsymbol{X}\right) = c + \sum_{j=1}^{d} g_j\left(X_j\right), \tag{3.29}$$

and several one-dimensional functions are estimated instead of one function of several variables.

The *Partial Linear Model* (PLM) splits explanatory variables into two groups: $\boldsymbol{X} = (\boldsymbol{U}, \boldsymbol{T})$ for analytical reasons or because of some assumptions of a given economic theory model. The regression of Y on $\boldsymbol{X} = (\boldsymbol{U}, \boldsymbol{T})$ takes the following form:

$$\mathbb{E}\left(Y|\boldsymbol{U}, \boldsymbol{T}\right) = \boldsymbol{U}^\top\beta + m(\boldsymbol{T}) \tag{3.30}$$

where $m(\cdot)$ us an unknown multivariate function of $\boldsymbol{T}$.

The *Generalized Additive Model* (GAM) introduces a known parametric link function $G(\cdot)$ to the *Additive Model* described above:

$$\mathbb{E}\left(Y|\boldsymbol{X}\right) = G\left\{c + \sum_{j=1}^{d} g_j(X_j)\right\}. \tag{3.31}$$

Analogously, the *Generalized Partial Linear Model* (GPLM) introduces a link $G(\cdot)$ to the *Partial Linear Model*:

$$\mathbb{E}\left(Y|\boldsymbol{U}, \boldsymbol{T}\right) = G\left\{\boldsymbol{U}^\top\beta + m(\boldsymbol{T})\right\}. \tag{3.32}$$

However, in contrast to the GAM, $m(\cdot)$ can be a multivariate nonparametric function.

Finally, given the problem of the curse of dimensionality and practical problem of interpretability when dealing with a nonparametric function $m(\cdot)$ in high dimensions, the *Generalized Partial Linear Partial Additive Model* (GAPLM) splits the nonparametric part of the GPLM into the sum of one-dimensional nonparametric functions as in the AM:

$$\mathbb{E}\left(Y|\boldsymbol{U}, \boldsymbol{T}\right) = G\left\{\boldsymbol{U}^\top\beta + \sum_{j=1}^{d} g_j\left(X_j\right)\right\}. \tag{3.33}$$

As one can see, non- and semiparametric models are a step forward from the classical parametric regression when no prior structure of the link between dependent and explanatory variables is assumed. However, for practical applications, it is frequent to deal with more than two explanatory variables, and therefore the direct application of nonparametric regression is undesirable in this case because of the curse of dimensionality and interpretability problems. Semiparametric models reduce the dimensionality of the problem, however at the cost of imposing extra structural assumptions, which can become a serious side-effect that neglects all other advantageous features when

misspecification of the model is severe.

Additionally, so far it was assumed that the set of explanatory variables $\boldsymbol{X}$ is known in advance. That may be true for some applications, but for lots of others (including stock picking) this is not the case because there is no exhaustive list of factors that definitely forms the DGP of stock yields available. One of the possible solutions is rolling specifications when different subsets of $\boldsymbol{X}$ are taken, different models are built, and their quality is assessed by the sample analog of the integrated square error, see Härdle et al. (2004) for more details. However, this approach, which closely resembles cross-validation in some sense, is rather slow by its nature and can not be recommended for the application to trading strategies where the execution time is critical.

Therefore, although non- and semiparametric regressions allow not to focus anymore on the functional specification of a regression, other aforementioned limitations suggest to switch to even more flexible modern statistical methods that are to be introduced below.

3.5 Decision Trees and Other Classification Methods

Classical parametric regression models require both explanatory variables and functional form of the regression to be precisely specified in advance. Sometimes this prerequisite does not seem to create any significant obstacles. For instance, in Fama and French (1988a) the linear link between dividend yields and expected stock returns was analyzed, and it is therefore totally fine to apply a linear parametric regression in such a setup. In many other practical applications like stock picking, where the structure of the data is not known in advance, that may become a more serious issue, see Fama (1998a) for a broader discussion on this topic. To cope with the unknown functional form, nonparametric regression theoretically offers an excellent solution, but because of the curse of dimensionality, less flexible semiparametric models are used in practice. However, even semiparametric models do not choose the proper explanatory variables from the sample automatically as they partially do with the functional form of the nonparametric portion of the regression. It may theoretically be possible to choose the best subset of variables based, for instance, on the mean squared error criterion of quality, but for practical applications, especially those which are critical to the execution time, this may not be a feasible solution.

Various classification methods may be flexible to a greater or lower extent in choosing the functional form of decision rules than the aforementioned regression techniques, but they are aimed at revealing unknown (and possibly nonlinear) data patterns and therefore should be seriously considered for the practical task of stock picking. Decision trees, as one will see below, go even further and perform automatic iterative input variable selection.

Recall that by now the dependent variable Y was the (unknown) next period stock price yield. If, for instance, the yield is measured against the market yield (replicated by some major stock index), then historical stock returns may therefore be classified as under- or overperforming (Sorensen et al., 1999). Stocks may be categorized into more performance buckets if needed, they can also be tagged based solely on individual performance, see Section 8.2 for more details. Therefore, the dependent variable Y can easily be transformed into a categorical one when necessary.

Perhaps one of the most natural methods of the discriminant analysis – a layer of classification techniques where the number of classes is known a priori – is the *maximum likelihood discriminant rule*. According to this rule, if there are J possible populations, which are distributed with the probability density functions $f_j(x)$, then an observation x is allocated to the population j that exhibits the maximum likelihood of containing this observation:

$$f_j(x) = \max_i f_i(x). \tag{3.34}$$

The practical application of this method to stock picking faces the same challenge of unknown multivariate density functions that may be quite difficult to estimate, see Section 3.4 for more details on non- and semiparametric estimation.

Let π_j be the prior probability of the population j. The *Bayes discriminant rule* expands the maximum likelihood discriminant rule in the following way: the observation x is allocated to the population j if

$$\pi_j f_j(x) = \max_i \pi_i f_i(x). \tag{3.35}$$

The Bayes rule is identical to the maximum likelihood discriminant rule for $\pi_j = \frac{1}{J}$ (Härdle and Simar, 2003).

The *Linear Discriminant Analysis* (LDA) is a group of methods that finds a linear combination of attributes such that it maximally separates two or more classes. If $Y = \boldsymbol{X}\omega$ is a linear combination of observations, then the Fisher's linear discrimination function (Fisher, 1936) selects the projection vector ω^* that keeps the between-class distance as large as possible while the within-class scatter should be as small as possible:

$$\omega^* = \max_\omega \frac{\omega^\top \boldsymbol{B}\omega}{\omega^\top \boldsymbol{W}\omega} \tag{3.36}$$

where

$$\boldsymbol{B} = \sum_c n_c(\mu_c - \bar{x})(\mu_c - \bar{x})^\top, \tag{3.37}$$

$$\boldsymbol{W} = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^\top, \tag{3.38}$$

and n_c is the number of cases in the class c, n – number of observations in the sample, $\mu_c = \frac{1}{n_c}\sum_{i \in c} x_i$, and $\bar{x} = \frac{1}{n}\sum_i x_i$, see Härdle and Simar (2003) for more details.

Although the LDA does not require density functions to be estimated, its final linear decision rule may not be flexible enough when classifying complex nonlinear structures. It is also worth pointing out that the decision rule is the same for any point to classify, see Section 3.3 for a broader discussion of this issue.

The *Quadratic Discriminant Analysis* (QDA) separates observations by a quadratic surface:

$$y = x^\top \boldsymbol{A}x + b^\top x + c \tag{3.39}$$

where x is a new observation to classify, $\boldsymbol{A}$, b, and c are parameters to estimate. Although the QDA provides potentially a more flexible decision rule, the cost is the increased number of parameters to estimate, see more on this in Hastie et al. (2001).

One of the simplest nonparametric classifiers, the *k-nearest neighbor* algorithm (k-NN), first introduced by Fix and Hodges (1989), is based on a distance (or dissimilarity)

measure d that is assigned to all pairs of observations that in the simplest case takes the form of the L_1 metric:

$$d(x, y) = \sum_i |x_i - y_i| \tag{3.40}$$

where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ is an arbitrary pair of observations.

Given k, the set of closest observations is then obtained, and the new observation x is assigned to the class label that appears most frequently in the domain. The parameter k is usually selected via cross-validation, see Section 5.2 for more details on this procedure.

It turns out that the 1-NN (k-NN with $k = 1$) classifier is the special case of the kernel density estimator (or the *Parzen window method* introduced in Parzen (1962)) when the kernel is selected accordingly, see Patrick (1972) for more details. Because the k-NN classifier in fact employs the local density estimation (Patrick, 1972), in a multidimensional case the method suffers from the curse of dimensionality, which was described in Section 3.4, and therefore can not be recommended for multidimensional classification problems such as stock picking.

A more advanced method, called *Artificial Neural Networks* (ANN), is favored by some researches for its potential flexibility of recognizing nonlinear data patterns. Inspired by the architecture of the central nervous system and neurons, in neural network models neurons are connected together to form a network of nodes. The desired signal flow comes from varying strength of connections (or weights). Given a set of learning sample pairs (x, y), $x \in \boldsymbol{X}$, $y \in Y$, the network $f(x)$ is derived from a composition of functions $g_i(x)$ that can be defined as compositions of other functions:

$$f(x) = K\left(\sum_i w_i g_i(x)\right) \tag{3.41}$$

where $K(\cdot)$ is some predefined function (the hyperbolic tangent is used quite frequently), w_i are weights, and $g_i(x)$ are other functions.

A common example of the neutral network is the *perceptron classifier* (Rosenblatt, 1958) that transforms an n-dimensional input into a binary output y: $\mathbb{R}^n \to 0, 1$. In this case, the form of the network (3.41) simplifies to

$$f(x) = \sum_i w_i x_i = w^\top x. \tag{3.42}$$

The obtained value is compared with the threshold value θ, and the perceptron is excited if the sum is greater than this value. With the help of the step function

$$g(x) = \begin{cases} 1 & \text{if } a \geq \theta, \\ 0 & \text{if } a < \theta, \end{cases} \tag{3.43}$$

the perceptron can be represented as $\mathbf{I}_{[\theta,\infty)}(w^\top x)$ where $\mathbf{I}_{[\theta,\infty)}(a) = g(a)$.

Given the observations (x, y) from the learning sample, the aim of the next step is to find a function f (in the allowed class of functions) that matches the examples. This is usually achieved via minimizing the cost function C. A commonly used cost is the mean-squared error that tries to minimize the average error between the network's output, $f(x)$, and the target value y over all the example pairs: $C = \mathbb{E}\left[(f(x) - y)^2\right]$.

A well-known backpropagation algorithm (Rumelhart et al., 1986) for training neural networks is obtained when the mean-squared error is minimized using the *gradient descent* for multi-layered perceptrons, a more powerful modification of the standard linear perceptron in that it uses three or more layers of neurons with nonlinear activation functions, see Haykin (1998) for more details.

Although Artificial Neural Networks appear to be a powerful method capable of producing nonlinear decision rules, it exhibits, unfortunately, some severe pitfalls (Breiman, 1994). For instance, it is not entirely clear how many hidden layers (or functions g_i) one should employ. When more hidden layers are added, more parameters are to be estimated and there is a higher risk of overfitting. Furthermore, the gradient descent method finds local minima and hence the initial values of the parameters are quite important because it is uncommon to run the procedure many times to find the global minimum – it would take too much time. Finally, Artificial Neural Networks can be extremely difficult – if not impossible – to interpret. Although there have been some attempts to apply Artificial Neural Networks to stock price modeling (such as forecasting of a stock price index) – see Dutta et al. (2006) for a more detailed overview – the aforementioned challenges and trading strategy prerequisites listed in Section 3.1 motivate one to look for better alternatives.

Support Vector Machines (SVMs) are a set of other powerful nonlinear classification and regression techniques that are based on the margin maximization between data classes (Vapnik, 1995), where the margin is the distance between the hyperplanes bounding each class. The classifier function used by the linear SVM is a hyperplane symmetrically surrounded by a margin zone – SVMs simultaneously minimize the empirical classification error and maximize the geometric margin. Although the original hyperplane algorithm proposed by Vapnik was a linear classifier, applying the kernel trick (Aizerman et al., 1964) to maximum-margin hyperplanes introduced SVM-based nonlinear classifiers (Boser et al., 1992).

More formally, an ideal classification function f from a set of the available functions $\mathcal{F}$ is the result of minimization of the expected risk:

$$R(f) = \int \frac{1}{2}|f(x) - y| dP(x, y) \tag{3.44}$$

where the distribution $P(x, y)$ is assumed to be known. In practice, however, this distribution is unknown, leading to an ill-posed problem (Tikhonov and Arsenin, 1977). Although $R(f)$ can not be minimized directly, it is possible to estimate the Vapnik-Chervonenkis (VC) bound (Vapnik, 1995) that holds with a certain probability η:

$$R(f) \leq \hat{R}(f) + \phi\left(\frac{h}{n}, \frac{\ln(\eta)}{n}\right) \tag{3.45}$$

where $\hat{R}(f)$ is the empirical risk function

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}|f(x_i) - y_i|, \tag{3.46}$$

h is the VC dimension, and the term $\phi\left(\frac{h}{n}, \frac{\ln(\eta)}{n}\right)$ introduces a penalty for the excessive

complexity of a classifier function.

The basic idea of the SVM classification is to find such a separating hyperplane that corresponds to the largest possible margin between the points of different classes. The SVM classifier is obtained via an optimization procedure that introduces a penalty for misclassification. Let ξ_i be the classification error that is related to the distance from a misclassified point x_i to the bounding hyperplane. The objective function corresponding to the penalized margin maximization takes the following form:

$$\frac{1}{2}\|\omega\|^2 + C\left(\sum_{i=1}^{n}\xi_i\right)^v \tag{3.47}$$

where $\|\cdot\|$ is the Euclidean norm of a normal vector – a vector that it is perpendicular to the separating hyperplane, parameter C characterizes the generalization ability of the machine, and $v \geq 1$ is a positive integer controlling the sensitivity of the machine to outliers. The minimization of the objective function is performed conditionally with constraints

$$\begin{cases} y_i(x_i^\top \omega + b) \geq 1 - \xi_i, \\ \xi_i \geq 0 \end{cases} \tag{3.48}$$

where $f(x) = x^\top \omega + b$ is the classification score (original linear SVM version).

Practical applications as those in Huang et al. (2004), where a comparative study of the application of Support Vector Machines and Backpropagation Neural Networks for an analysis of corporate credit ratings was performed, suggest that SVMs and ANN perform comparably. Despite its popularity, however, SVMs have some drawbacks in certain situations (Wu and Liu, 2007). In particular, the SVM classifier can be very sensitive to outliers in the training sample. When there exist points far away from their own classes (namely, 'outliers' in the training data), the SVM classifier tends to be strongly affected by such points because of its unbounded hinge loss. Moreover, the number of support vectors (SVs) can be very large in many applications, especially for difficult classification problems or problems with numerous input variables. An SVM classifier with many SVs may require longer computation time, especially for the predication phase. The authors of the recently introduced Robust SVM (RSVM) (Wu and Liu, 2007) claim to tackle these challenges at least partially by delivering more robust classifiers and the fact that RSVM tends to use smaller and more stable set of SVs than that of the original SVM. In the meantime, however, classical SVMs seem to be an infeasible, although quite promising, nonlinear classification technique for stock picking task mainly due to the long computation time and ambiguity of choosing the input model parameters and explanatory variables.

To summarize briefly, what are the main modeling challenges arising from the potential application of the aforementioned methods to stock picking? One of the most important is the assumed functional form of the DGP forming the unknown next period stock return. While this could be a severe problem for general equilibrium models and parametric regression, a nonparametric regression model may seem to be the way out. However, due to the curse of the dimensionality, a nonparametric regression is hardly a feasible choice when there are many input variables in the model, and empirical evidence from Section 1 suggests exactly such a setup. Unfortunately, a semiparametric regression model solves the problem only partially – it effectively reduces the dimensionality

of the estimation problem, but only at the cost of introducing additional functional assumptions. But even having supposed that a certain semiparametric model does not impose too restrictive limitations, the question of selecting the input variables for the model would still be open.

This brings us to the classification layer of statistical techniques. However, some of them as the LDA are not flexible, others like the k-NN face the same limitations (the curse of dimensionality). SVMs may seem to be a perfectly flexible approach, however building classification rules for many stocks independently over long time periods implies enormous computation expenses for nonlinear SVMs. And yet again, the list of 'correct' input variables is supposed to be available in advance, which is hardly the case for almost any economic application.

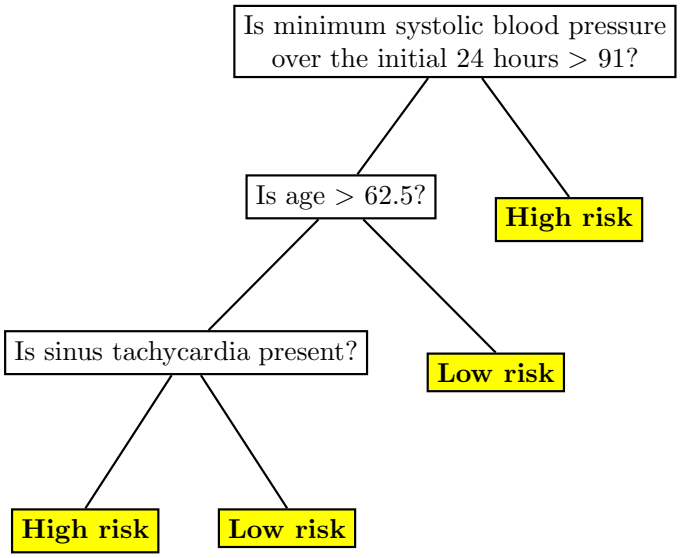

Figure 3.1: A medical application of the binary decision tree. Left branches stand for positive answers, right branches – for negative ones. Patients are classified into two groups of people having either high or low risk of not surviving at least 30 next days based on 19 various measured variables during the first 24 hours (Breiman et al., 1987).

Binary *classification and regression trees* (CART), a nonparametric method introduced in Breiman et al. (1987), seem to tackle the majority of the aforementioned challenges quite effectively. First of all, due to its nature (see Figure 3.1), the results are highly interpretable. Decision trees are a nonlinear classifier that is quite easy to implement computationally because the learning sample is split recursively by introducing filters in the form of 'yes-no' questions: Is $X_i < x$? Moreover, the method is robust to the effect of outliers and selects the input variables from the sample automatically. As one can see, most of the drawbacks of other statistical and econometric methods, briefly reviewed above, are managed quite effectively due to the special architecture of the method. Decision trees perform the classification by dividing orthogonally the data space so that a split reduces the variance in each of the subspaces and maximizes the variance between them.

Because CART is a nonparametric classification method that avoids explicit density estimation (and therefore, the curse of dimensionality), can reveal complex nonlinear structures, is computationally efficient, chooses the input variables automatically, and offers excellent interpretability, this classification method offers substantial advantages compared to other reviewed modeling methods, and that is the major reason for this technique to be employed for the stock picking core of the trading algorithm presented in the study.

4 Introduction to Binary Classification Trees

4.1 What is a Classification Tree?

Binary classification trees are a nonparametric method of data classification introduced in Breiman et al. (1987). One of its peculiarities is the special form of produced decision rules – binary decision trees. These trees are constituted by *nodes*, and each node carries a 'yes-no' question in the form $X_i \leq x$? where X_i is one of the explanatory variables (features) and x is the question value. Both X_i and x are chosen automatically by CART as well as the tree size, i.e. the number of necessary filters and their configuration. When new data are to be classified, they are processed by sequential posing of tree questions: left branches stand for positive answers and right branches – for negative ones. Each node of a tree in the bottom has a class tag, in this way classified data are assigned to one of the predefined groups. The type of nodes in the bottom is called *terminal*.

Figure 4.1 introduces a simple two-dimensional data structure. Its observations are of one of five predefined classes, which are marked with different colors. Each split clearly separates one homogenous data cluster that constitutes a terminal node with a respective class tag.

Decision trees can be created from the available data, e.g. data from the past. If a certain link between some objects is assumed, then the first step to build a tree is to create a *learning sample*. In the framework of stock picking, future stock price fluctuations are assumed to be driven by present changes of fundamental or technical company indicators like Earnings Per Share. Then factors similar to Earnings Per Share (Cash Flow, Return on Equity, Sales, etc.) are grouped into explanatory variable set $\boldsymbol{X} \in \mathbb{R}^p$ (where p is the overall number of explanatory factors) while the target characteristic – the next period stock price yield – is defined by the class vector Y. The natural range of values $y \in Y$ in this particular case is $\{long, short, neutral\}$ standing for undervalued, overvalued, and fairly priced stocks respectively.

A learning sample, frequently built using available historical data, therefore contains a set of observed market situations $(\boldsymbol{X}, Y)$. The goal of the decision tree as a classification technique is to reconstruct the (possibly nonlinear) link between $\boldsymbol{X}$ and Y in the form of a binary classification tree, which later can be applied to classify new data (i.e. current company data, for instance) to assess its target characteristic (for instance, the next period company's stock price movement).

The application of decision trees to a data set with observations of an unknown class implies three major steps to be conducted:

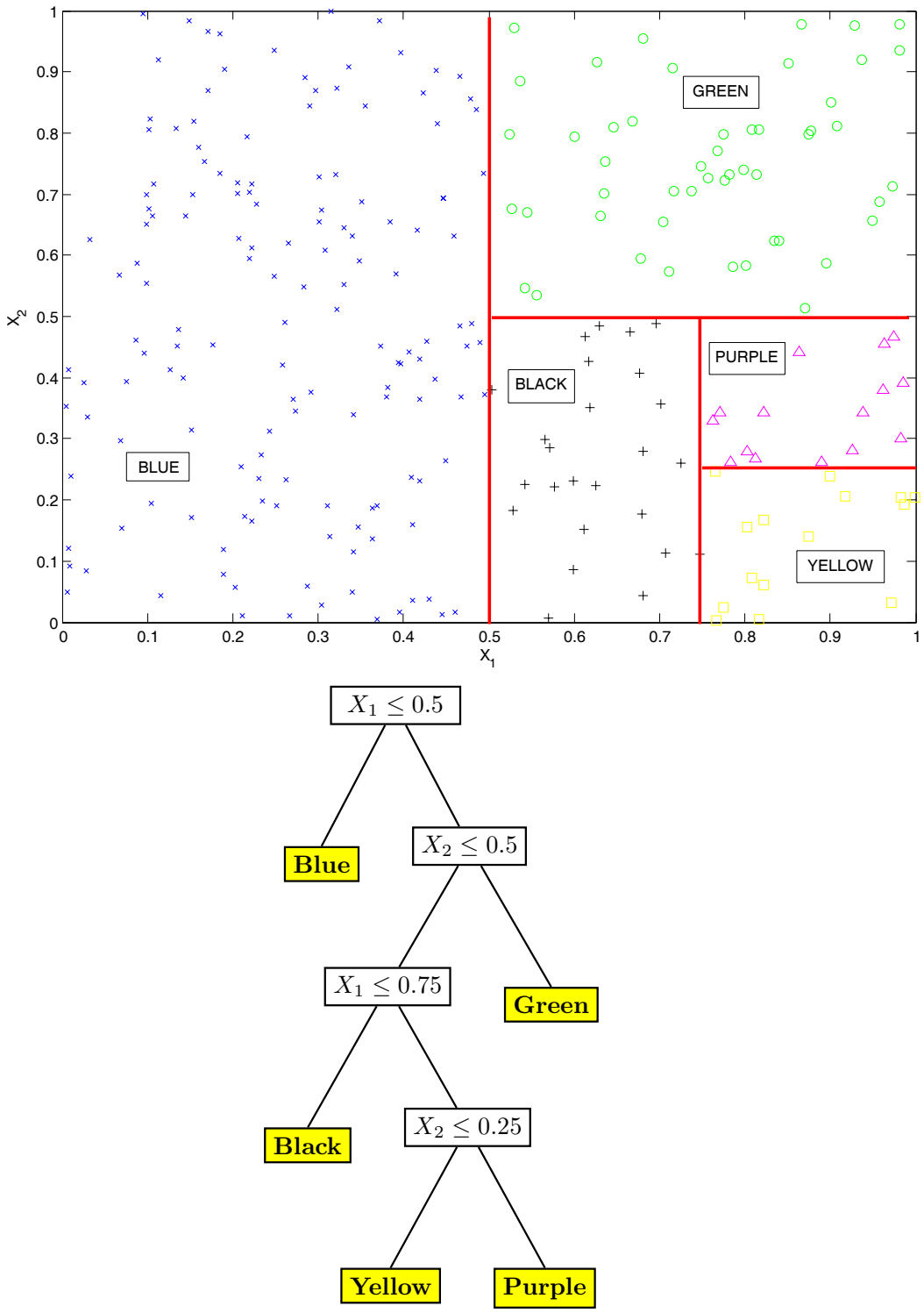

Figure 4.1: Application of CART to an artificial two-dimensional data set. The *root node* at the top contains a filter $X_1 \leq 0.5$. There are five terminal nodes in this tree and five classes: *blue*, *green*, *black*, *yellow*, and *purple*. Left branches stand for positive answers, rights ones – for negative answers

- construction of the so called *maximum tree* T_{MAX} (see below),
- choice of the right tree size (tree pruning) T^*,
- classification of new data using the constructed tree T^*.

A maximum tree is the one containing observations of the same class at each of the terminal nodes. The *root node* – the one at the top of any tree – resembles the whole learning sample. Moving from the root node down the tree, the learning sample is being split recursively in a way that more homogenous clusters of observations are separated into tree nodes. This can be achieved as follows.

4.2 Impurity Measures for Classification Trees

Suppose there are n observations in the learning sample and n_j is the overall number of observations belonging to the class j, $j = 1, \ldots, J$. In the described stock picking setup where $y \in \{long, short, neutral\}$, J is equal to three. Then define *class probabilities* as following:

$$\pi_j = \frac{n_j}{n}, \tag{4.1}$$

i.e. a proportion of observations belonging to a particular class relative to the overall number of observations.

Let $n(t)$ be the number of observations in the node t and $n_j(t)$ – the number of observations belonging to j-th class in the same node t. Then the joint probability of the event that an observation of the j-th class falls into the node t is

$$p(j,t) = \pi_j \frac{n_j(t)}{n_j}. \tag{4.2}$$

Hence $p(t) = \sum_{j=1}^{J} p(j,t)$, and the *conditional probability* of an observation to belong to the node t given that its class is j is computed as following:

$$p(\,j\,|\,t) = \frac{p(j,t)}{p(t)} = \frac{n_j(t)}{n(t)}, \tag{4.3}$$

which is a proportion of the class j in the node t. One can easily show that $\sum_{j=1}^{J} p(\,j\,|\,t) = 1$.

A measure of a classification tree that shows the degree of class heterogeneity in a given node is called an *impurity measure* $i(t)$. With its help, different node configurations or *splits* – combinations of question variables X_i and question values x – can be compared, and therefore only one of them can finally be selected to be incorporated in the tree, thus making it possible to determine specific *optimal* (see below) values of X_i and x employed in the questions. An impurity measure can be defined via an *impurity function* $\varphi(t)$ that is determined on the subsets $\{p_1, \ldots, p_J\}$ for $\forall j$ and $p_j \geq 0$, $j = 1, \ldots, J$, $\sum_{j=1}^{J} p_j = 1$ so that

1. $\varphi(\cdot)$ has a unique maximum at the point $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$;
2. $\varphi(\cdot)$ has a unique minimum at points $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, …, $(0, 0, \ldots, 1)$;
3. $\varphi(\cdot)$ is a symmetric function of class probabilities $p_1, \ldots, p_J$.

Each function satisfying these conditions can be called an impurity function. Functions taking only non-negative values will be considered in this study. Given the function $\varphi(\cdot)$, define an impurity measure $i(t)$ for a node t as

$$i(t) = \varphi\left(p(1|\,t), p(2|\,t), \ldots, p(J|\,t)\right). \tag{4.4}$$

It is important to point out that from given definitions it follows that it is possible to define *multiple* impurity measures for the same node t.

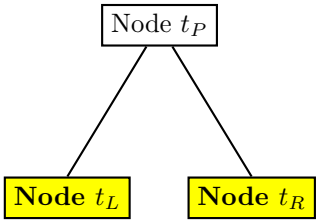

Figure 4.2: The triplet of nodes: t_P – parent node, t_L – left child node, and t_R – right child node

Let t_P be the parent node and t_L, t_R – left and right child nodes of the parent node t_P respectively so that a fraction p_L of observations from the node t_P follows to the left child node, and a fraction $p_R = (1 - p_L)$ – to the right one.

If n_P is the number of observations in t_P and n_L, n_R – in t_L and t_R respectively, then the probabilities of an observation to fall into one of two child nodes can be computed as following:

$$p_L = \frac{n_L}{n_P}, \quad p_R = \frac{n_R}{n_P}. \tag{4.5}$$

Denote an arbitrary data split by s. A functional that determines the question at each tree node – split s^* – is the maximum value of the one-level decrement of the impurity function $i(s, t_P)$, which can be computed for an arbitrary node t_P:

$$\Delta i(s, t_P) = i(t_P) - p_L \cdot i(t_L) - p_R \cdot i(t_R). \tag{4.6}$$

Obviously, the higher is the value of $\Delta i(s, t_P)$ – the better split has been obtained since it was possible to reduce data impurity more significantly. Since $t_L \cup t_R = t_P$, the value $\Delta i(s, t_P)$ represents a change of data impurity in t_P solely due to the split s.

To find the optimal s for a given node, it is natural to maximize $\Delta i(s, t_P)$ by different s, i.e. choosing different variables from the learning sample and adjusting the relevant question values. In this way, a classification tree of any configuration up to a maximum tree can be built.

While searching for the optimal value of s^*, the value of $i(t_P)$ remains constant because it does not depend on X_i and x that together create t_L and t_R. Hence, it is equivalent

to state that

$$s^* = \operatorname{argmax}_s \Delta i(s, t_P) = \operatorname{argmax}_s \{-p_L i(t_L) - p_R i(t_R)\} = \\ = \operatorname{argmin}_s \{p_L i(t_L) + p_R i(t_R)\} \tag{4.7}$$

where t_L and t_R are implicit functions of s.

If resulting nodes t_L and t_R are not enough class homogenous (see below), the same procedure can be looped until a decision tree becomes of a required configuration. Classes are then assigned to terminal nodes using the following rule:

$$\text{If } p(j|\,t) = \max_i p(i|\,t), \text{ then } j^*(t) = j. \tag{4.8}$$

If the maximum is not unique, then the class $j^*(t)$ is assigned arbitrary from the pool of arguments $\{i\}$ for which $p(i|\,t)$ takes its maximum value.

Note that the criterion in (4.7) can be used when any valid impurity measure is employed there, therefore making the whole algorithm quite versatile.

It may, though, have a little drawback that is worth pointing out. Maximizing the decrement of the impurity function means that only two levels of a decision tree are taken into account whereas other parts of the tree (such as possible child nodes of t_L and t_R) can not influence the choice of the optimal split. That is why the procedure can be characterized only as *locally optimal*.

Is it possible to build a *globally optimal* algorithm of data splitting? One of the potential criteria that takes into account the whole structure of the tree could be the *integral tree impurity* decrement that is a weighted sum of local tree impurity decrements.

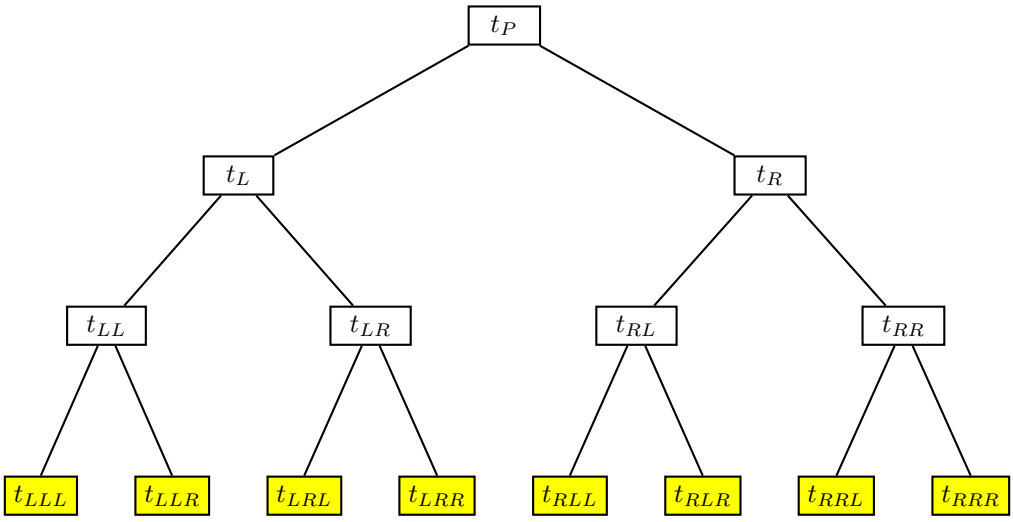

Figure 4.3: A maximum binary decision tree containing three splitting levels, $M = 3$.

Figure 4.3 displays the first three splitting levels of an arbitrary decision tree. For each splitting level, let us unite into groups node probabilities and corresponding node impurities. At the first level, where t_P is split into t_L and t_R, there are two corresponding

probabilities and impurities:

$$\begin{cases} p_L^{(1)} = p_L, \quad p_R^{(1)} = p_R \\ i_L^{(1)} = i(t_L), \quad i_R^{(1)} = i(t_R) \end{cases} \tag{4.9}$$

Therefore, the conventional locally optimal impurity decrement rule can be rewritten as

$$\Delta i^{(1)} = \Delta i(s, t_P) = i(t_P) - \left\langle p_L^{(1)}, i_L^{(1)} \right\rangle - \left\langle p_R^{(1)}, i_R^{(1)} \right\rangle \to \max_s \tag{4.10}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product.

Due to the binary nature of a tree, deeper splitting levels contain more elements. For the next two levels, the groups will contain the following elements:

$$\begin{cases} p_L^{(2)} = \{p_{LL}, p_{RL}\}, \quad p_R^{(2)} = \{p_{LR}, p_{RR}\} \\ i_L^{(2)} = \{i(t_{LL}), i(t_{RL})\}, \quad i_R^{(2)} = \{i(t_{LR}), i(t_{RR})\} \end{cases} \tag{4.11}$$

and

$$\begin{cases} p_L^{(3)} = \{p_{LLL}, p_{LRL}, p_{RLL}, p_{RRL}\}, \\ p_R^{(3)} = \{p_{LLR}, p_{LRR}, p_{RLR}, p_{RRR}\}, \\ i_L^{(3)} = \{i(t_{LLL}), i(t_{LRL}), i(t_{RLL}), i(t_{RRL})\}, \\ i_R^{(3)} = \{i(t_{LLR}), i(t_{LRR}), i(t_{RLR}), i(t_{RRR})\} \end{cases} \tag{4.12}$$

If a node contains only observations of the same class (and thus the impurity measure reaches its minimum according to the definition of the measure), then its child nodes are empty sets and

$$\begin{cases} p_L(\emptyset) = p_R(\emptyset) = 0, \\ i(\emptyset) = 0. \end{cases} \tag{4.13}$$

The globally optimal impurity decrement rule is proposed in the following form:

$$\Delta i^{(M)} = i(t_0) - \sum_{m=1}^{M} \left(\left\langle p_L^{(m)}, i_L^{(m)} \right\rangle - \left\langle p_R^{(m)}, i_R^{(m)} \right\rangle \right) \to \max_{s^{(1)}, \ldots, s^{(M)}} \tag{4.14}$$

where M is the biggest splitting level achieved for the maximum tree and $\left\{ s^{(1)}, \ldots, s^{(M)} \right\}$ are all possible splits for each split level of a tree.

As it can be seen from (4.14), bigger chunks of data influence the final splitting configuration more due to the higher values of respective probabilities. The whole tree is created at once as opposed to (4.10) where only a single split is determined at a time. Note that a locally optimal decision tree, i.e. one built via (4.10), is not (under general conditions) globally optimal. Quite interestingly, a globally optimal tree may be not locally optimal.

Unfortunately, because of the enormous computing power required to build an optimal tree via (4.14), it is not (yet) practically possible to test its relative efficiency. However, the rapid development of microprocessors provides serious reasons to conclude that this and other resource-hungry algorithms will become practically feasible in the near future.

In the financial sphere where computations are sometimes required to be carried out virtually online or at least to be conducted very quickly, the speed matter becomes crucial, that is why it is reasonable to apply locally optimal procedures. Once the

enhanced precision is required, e.g. in credit scoring or portfolio optimizations where the computation speed recedes into the background, then if the sufficient computing power is available, it is possible to construct globally optimal decision rules.

At the time, in this work the locally optimal variant is used.

4.3 Two Special Functional Forms of Impurity Measures

Perhaps one of the most natural ways to define data impurity is to use the *variance* measure. Breiman et al. (1987) introduce it as following: assign 1 to all observations at the node t belonging to the class j and 0 to the others. Then the sample variance estimate for node t observations is

$$p(j|\,t)\,(1-p(j|\,t)\,. \tag{4.15}$$

Summing over all J classes, one obtains the so called *Gini index*:

$$\sum_{j=1}^{J} p(j|\,t)\,(1-p(j|\,t) = 1-\sum_{j=1}^{J} p^2(j|\,t). \tag{4.16}$$

Thus the Gini index can be considered as a function $\varphi(p_1,\ldots,p_J)$, which in its turn is a second degree polynomial with non-negative coefficients. For each convex function it holds that for $\forall\alpha \geq 0$:

$$\begin{aligned}\varphi\left(\alpha p_1+(1-\alpha)p'_1,\ \alpha p_2+(1-\alpha)p'_2,\ \ldots,\ \alpha p_J+(1-\alpha)p'_J\right) > \\ > \alpha\varphi(p_1,\ldots,p_J)+(1-\alpha)\varphi(p'_1,\ldots,p'_J).\end{aligned}$$

Since

$$\begin{aligned}\varphi(\alpha p_1+(1-\alpha)p'_1,\ldots,\alpha p_J+(1-\alpha)p'_J) = \left[1-\alpha\sum_{j=1}^{J}p_j^2\right]+ \\ +\left[1-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2\right] = 2-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2\end{aligned}$$

and

$$\begin{aligned}\alpha\varphi(p_1,\ldots,p_J)+(1-\alpha)\varphi(p'_1,\ldots,p'_J) = \left[\alpha-\alpha\sum_{j=1}^{J}p_j^2\right]+ \\ +\left[(1-\alpha)-(1-\alpha)\sum_{j=1}^{J}{p'}_j^2\right] = 1-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'}_j^2,\end{aligned}$$

one can conclude that

$$2-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'}_j^2 > 1-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'}_j^2.$$

Hence the function $\varphi(\cdot)$ in the form of the Gini index is convex.

This property of the Gini index is quite important. Since $\varphi(p_1, \ldots, p_J)$ is a convex function and $p_L + p_R = 1$, the following inequality holds:

$$\begin{aligned} i(t_L)p_L + i(t_R)p_R &= \varphi\left(p(1|\,t_L), \ldots, p(J|\,t_L)\right)p_L + \varphi\left(p(1|\,t_R), \ldots, p(J|\,t_R)\right)p_R \\ &\leq \varphi\left(p_L p(1|\,t_L) + p_R p(1|\,t_R), \ldots, p_L p(J|\,t_L) + p_R p(J|\,t_R)\right) \end{aligned}$$

where the inequality becomes the equality in the case when $p(j|\,t_L) = p(j|\,t_R)$, $j = 1, \ldots, J$.

Recall that

$$\frac{p(j, t_L)}{p(t)} = \frac{p(t_L)}{p(t)} \cdot \frac{p(j, t_L)}{p(t_L)} = p_L \cdot p(j|\,t_L)$$

and since

$$p(j|\,t) = \frac{p(j, t_L) + p(j, t_R)}{p(t)} = p_L p(j|\,t_L) + p_R p(j|\,t_R),$$

one can conclude that

$$i(t_L)p_L + i(t_R)p_R \leq i(t). \tag{4.17}$$

Hence each variant of the data split leads to $\Delta i(s, t) > 0$ unless $p(j|\,t_R) = p(j|\,t_L) = p(j|\,t)$, i.e. when even the best available univariate filter can not decrease class heterogeneity.

Given the way how the Gini index is computed, it becomes obvious that this impurity measure can be quite effective. First, it is easy and fast to compute and second, as it was mentioned above, the Gini index is relatively robust to the effect of outliers – a few outliers can not drastically change the values of $p(j|\,t)$, $j = 1, \ldots, J$, hence s^* is not affected.

But, of course, the impurity measure can be defined in a number of different ways. For various practical applications a so called *twoing rule* can also be considered.

Its idea is completely different compared to the Gini index. Instead of looking for maximization of the impurity measure decrement at a particular node, the twoing rule tries to balance the constructed tree in a special way as if the learning sample contained only two classes. The reason for such an algorithm is that a decision rule based on the twoing criterion would be able to distinguish observations between general factors on top levels of the tree and take into account specific data characteristics at lower levels.

If $S = \{1, \ldots, J\}$ is a set of learning sample classes, let us divide it into two subsets

$$S_1 = \{j_1, \ldots, j_n\}, \text{ and } S_2 = S \backslash S_1$$

so that all observations belonging to S_1 get dummy class 1, and the rest – dummy class 2.

The next step is to calculate $\Delta i(s, t)$ for different s *as if there were only two dummy classes*. Since actually $\Delta i(s, t)$ depends on S_1, the value $\Delta i(s, t, S_1)$ is maximized. That is why a *two-step procedure* is obtained: first, find $s^*(S_1)$ maximizing $\Delta i(s, t, S_1)$ and second, find a *superclass* S_1^* maximizing $\Delta i(s^*(S_1), t, S_1)$.

In other words, the idea of the twoing criterion is to find such a combination of superclasses at each node as if the impurity measure decrement was maximized for the data only with two classes $S = \{1, 2\}$.

This method provides one big advantage – it finds the so called *strategic nodes*, i.e. nodes that filter observations in the way they are different to the maximum feasible extent.

The application of the twoing rule in practice may seem to be particularly suitable for data with a big number of classes. However, it is exactly for that kind of data a substantial challenge, namely computation speed, can arise. Assume that the learning sample has J classes, then a set S can be split into S_1 and S_2 in 2^{J-1} ways. For 11 classes, the data in the learning sample will create more than 1000 combinations. Fortunately, there is a theoretical result helping to reduce drastically the amount of necessary computations.

It can be proven (Breiman et al., 1987) that in a classification task with two classes and the impurity measure $p(1|t)p(2|t)$ for an arbitrary split s a superclass $S_1(s)$ is determined as following:

$$\begin{cases} S_1(s) = \{j : p(j|t_L) \geq p(j|t_R)\}, \\ \max\limits_{S_1} \Delta i(s,t,S_1) = \frac{p_L p_R}{4} \left[\sum\limits_{j=1}^{J} |p(j|t_L) - p(j|t_R)|\right]^2. \end{cases} \tag{4.18}$$

Hence, the twoing rule can be easily applied in practice as well as the Gini index, although the first criterion works a bit slower.

4.4 Gini Index and Twoing Rule in Practice

The two presented impurity measures could be illustrated by an example using the same data set and building two decision trees – one via the Gini index and another via the twoing rule.

Let us consider the data set with 400 observations characterizing various automobiles: their make, type, color, technical parameters, age, etc. (*Salford Systems*). The aim of this example is to build a decision tree splitting different cars by their makes based on other available variables from the sample.

A particular feature of the tree produced via the Gini index (Figure 4.4) is that at each node observations belonging to one make are filtered, i.e. observations with the most striking characteristics are separated. As a result, a decision tree is able to pick out automobile makes quite easily. As one can see, each displayed spilt of a tree produced by the Gini index contains one terminal node with a specific make. Therefore, the Gini index tends to select some features that are specific for a given single class and for a group of classes.

The next tree on Figure 4.5, which is built via the twoing rule, looks somewhat different, although the underlying data sample remains the same. Instead of specifying particular car makes at each node, the application of the twoing rule results in the demonstration of strategic nodes, i.e. questions that distinguish observations between different car classes to the maximum extent. Terminal nodes no longer contain cars of a specific make, but instead – pool of cars belonging to the same functional, price, size, or other group.

This feature can be vital when high-dimensional data sets with a big number of classes are processed. A typical example here could be the speech recognition problem – every

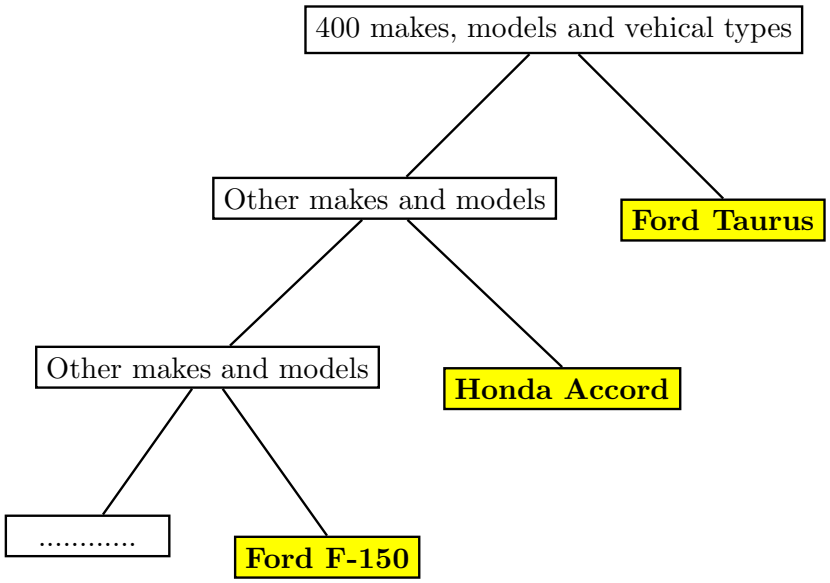

Figure 4.4: Classification rule yielded by the Gini index as the impurity measure

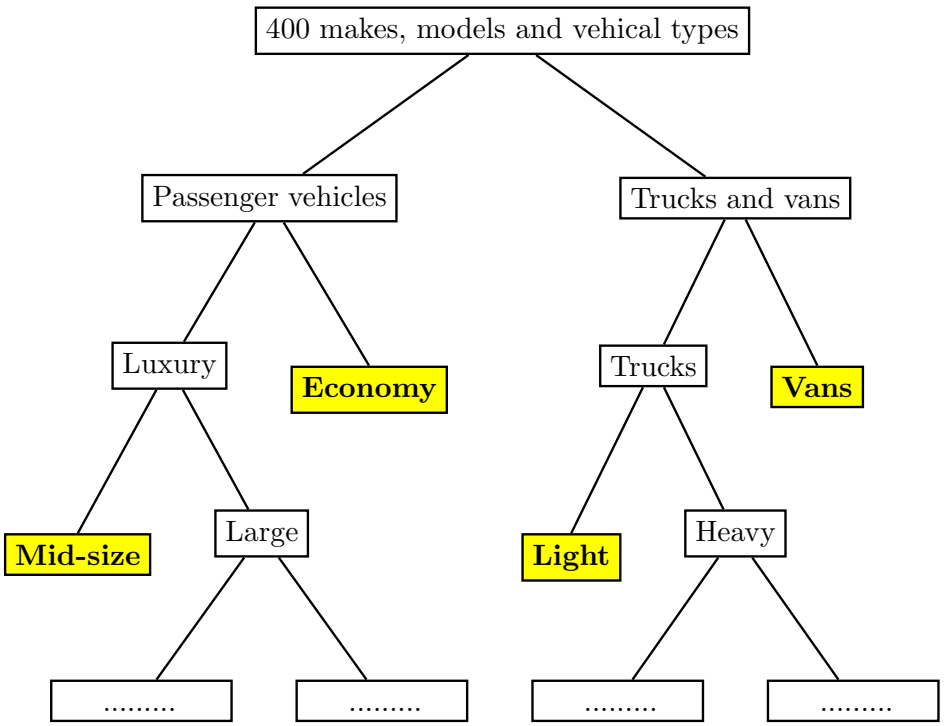

Figure 4.5: Classification tree constructed by the twoing rule

word can be coded with a new class, then if the twoing rule is applied, a classification tree is supposed to split different words by the number of syllables: with one and more than one syllables. At the next step other similar words' characteristics will be probably taken into account.

In Kolyshkina and Brookes (2002) the Gini-criterion based model produced only slightly better results in terms of prediction accuracy and ranking than that of the twoing criterion-based model. Moreover, as it was mentioned above, the typical values of the dependent variable for the stock picking task are $y \in \{short, neutral, long\}$, and the number of classes is likely to be very limited even if classes are assigned on a different basis, creating no extra motivation to use the twoing index. Therefore, at at this point of time, the Gini index, as one of the most frequently used impurity measures, would be preferred when constructing decision trees for the following part of this work dedicated to DAX30 stock picking.

4.5 Other Tree Induction Measures

Apart from the Gini index and the twoing rule, other impurity measures, which do not violate conditions listed in Section 4.2, can certainly be employed. One of such measures is entropy, see Cheng et al. (1999) and Simovici et al. (2000) for more details on its applications to numerical and categorical data respectively. Section 6.1 focuses on alternative tree induction procedures and considers more complex decision tree types.

However, as it will be shown below, apparently the impurity measure is an important, but not a crucial step in the decision tree optimization. Similar to other classification and regression techniques, decision trees can be overfitted, i.e. when their size (or, put differently, the number of terminal nodes) is unreasonably big, and a classification rule is no longer describes only the fundamental link between dependent and independent variables, but is influenced greatly by other factors of stochastic nature. A tree, therefore, is usually to be pruned, i.e. its size is to be balanced, see Chapter 5 for more details.

5 Cost-Complexity Tradeoff as a Traditional Way of Finding Optimal Tree Size

5.1 Early Stopping Rules

Although it is possible to grow a maximum tree for a given learning sample using (4.7) sequentially, the direct application of such a tree for the classification of new data is far not always desirable because of frequent *overfitting* – a situation when the training error reaches zero, but the validation error is usually much greater than its minimum level, which is feasible with a smaller tree. Note, however, that for some rare examples like on Figure 4.1 the direct application of the maximum tree is not undesirable – T_{MAX} is the best choice there.

For the majority of other cases, though, the problem of over- or underfitting can be severe. T_{MAX} can separate observations in such a way that correct class tags are attributed to each point when performing the *in-sample* classification, i.e. classification of observations that have been employed to build the classification rule. On the one hand, bigger and more complicated trees provide the ability to separate 'undesirable' observations (e.g. outliers) into local nodes. Growing a bigger and bigger tree, one can eventually get a decision rule where the impurity of each terminal node is close to or equals zero. However, such nodes, possibly containing only a few observations, may be indistinguishable from the counterparts containing mostly outliers. The reason is that the maximum classification tree accounts for *any*, even small and insignificant, data variations that can be caused by random shocks or measurement errors.

That is why when an unclassified observation is processed using the maximum tree, with a high probability it can fall into a terminal node describing such kind of disturbance. Hence the recommendation of a rule may be biased due to its excessive complexity – put differently, when the tree is overfitted. That is why the maximum tree tends to explain purely random effects using the factor space of the learning sample, especially when the terminal nodes contain only a few observations each. But such an explanation is usually only spurious, moreover – there is no guarantee that future random disturbances can be successfully accounted in the same way. There is a small probability that when classifying a new observation, it falls into the 'fundamental' part of the tree.

Therefore, too complex decision trees have high chances to be overfitted. On the other hand, a too small or simple decision rule is not a panacea. In this case, significant relationships probably could not be revealed since only a few iterations were used to split the learning data set – a typical situation of *underfitting*. Underfitted decision rules tend to be too rough and possibly do not account for some fundamental data links.

Obviously, some tradeoff between too simple and too complex decision rules is re-

quired. One way to achieve the reasonable value of the validation error could be the employment of some kind of an *early stopping rule*. Recall that the size of a tree is determined by the number of times the criterion (4.7) is employed – unless the maximum tree is grown. In (4.7) the value of $\Delta i(t, s)$ is maximized sequentially, and since the growth of a tree is controlled by the decrement of an impurity function, the following criterion could be introduced to stop expanding the tree size:

$$\Delta i(t_P, s^*) < \bar{\beta} \tag{5.1}$$

for some $0 < \bar{\beta} < 1$. It is worth pointing out that the maximum tree can be built applying the rule (5.1) when $\bar{\beta} = 0$.

However, $\Delta i(\cdot)$ is usually a non-monotone function of the tree size (Breiman et al., 1987), therefore a signal to stop could be premature.

This statement can be illustrated with a decision tree built for the BMW stock, refer to Figure 5.1, where a tree close to the maximum one is depicted. The impurity measure and its decrements can be computed at each splitting point. Because of the binary nature of the tree, the dynamics of the values of $\Delta i(s^*, t)$ can be traced once a certain path is followed from the root node to a given terminal node. Such path is marked with dashed lines on Figure 5.1. Using (4.6) and (4.16), the values of $i(t, s^*)$ and $\Delta i(t, s^*)$ were computed and plotted on Figure 5.2 and Figure 5.3 respectively.

As one can easily see, both $i(t, s^*)$ and $\Delta i(t, s^*)$ are non-monotone functions of the split level, therefore a general rule for setting a specific stopping value of $\bar{\beta}$ will eventually fail – initial splits (at the root node and its children) produce lower impurity decrement values than those achieved closer to terminal nodes!

Another possible way to determine the adequate shape of a decision tree is to set up a restriction on the minimum number of observations $\bar{N}$ at each terminal node. If at the terminal node t the number of observations is greater than requested:

$$n(t) > \bar{n}, \tag{5.2}$$

then splitting continues since the data are still not supposed to be clustered well enough.

Note that unlike the dynamics of $\Delta i(t, s)$, the dynamics of $n(t)$ is *always monotone* due to the way binary trees are created: each splitting question with a positive underlying value of $\Delta i(t, s^*)$ leads to filtering of at least one observation, therefore decreasing the value of $n(t)$ for child nodes.

However, the problem of estimation of $\bar{n}$ still exists. This absolute value would certainly depend on the overall sample size, and it might also be different for different applications.

Moreover, each early stopping rule like the two described here eventually produce the final optimal decision rule without analyzing the whole data structure. Setting specific values of $\bar{\beta}$ and $\bar{n}$, *a posteriori* – when the maximum tree is available for comparison – it may become apparent that the selected values are not optimal. More sophisticated – and frequently more efficient criteria – employ the maximum tree as the starting point to be able to analyze the whole data structure and prune it upwards ensuring the best (in some specific sense) tradeoff between complexity of a resulting tree and its predictive power.

Cross-validation is an example of such a procedure, which is to be described in the

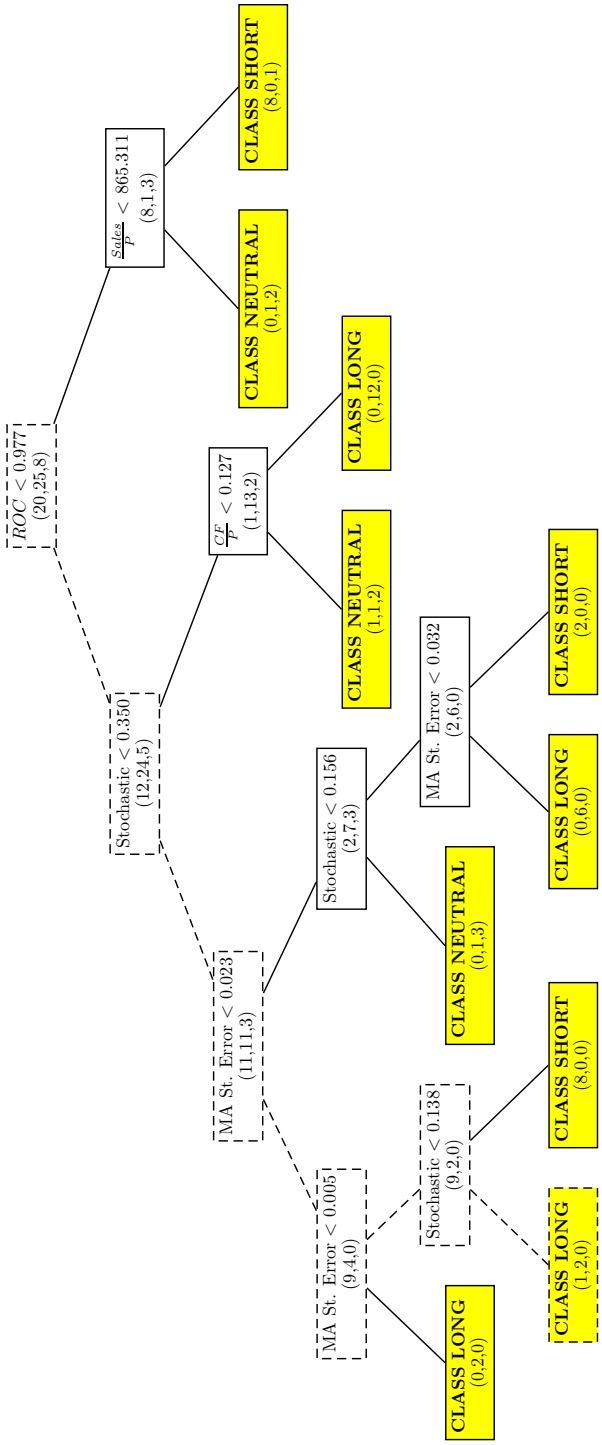

Figure 5.1: Decision tree example – BMW stock, see Section 8.3 for variable description. Dashed part of the tree marks the path for the impurity measure example on Figure 5.2 and Figure 5.3. Numbers in parentheses represent the amount of observations in a given node belonging to classes *short*, *long*, and *neutral* respectively

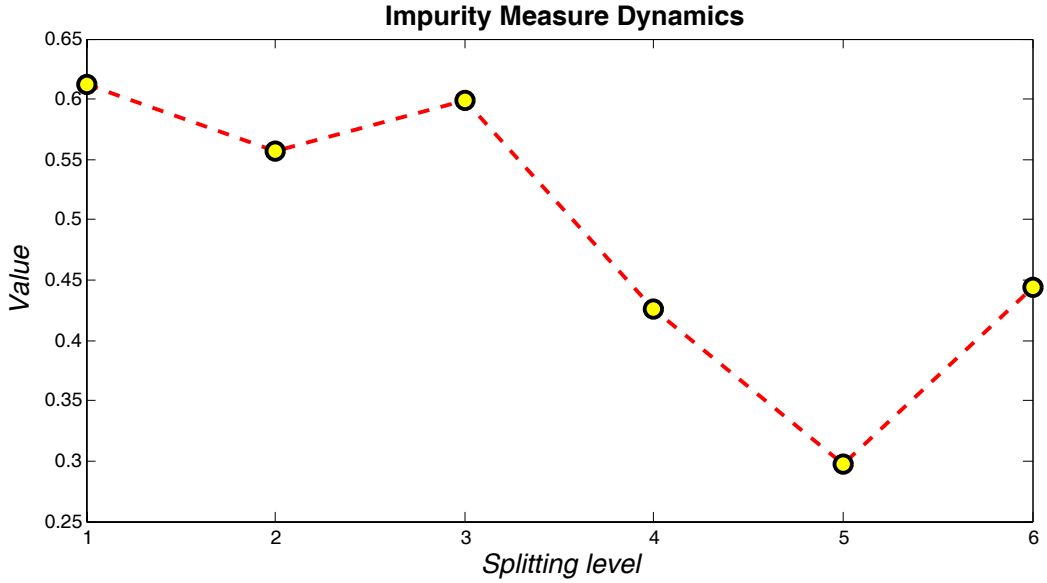

Figure 5.2: Impurity measure values $i(s^*, t)$ computed for the dashed part of the tree on Figure 5.1 – from the root node to the terminal node

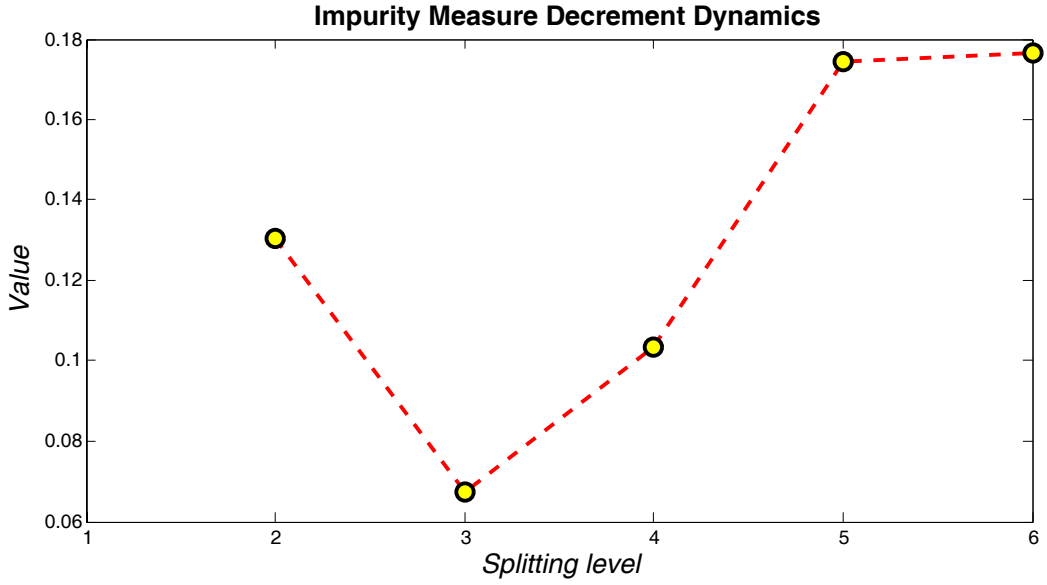

Figure 5.3: Respective values for the impurity measure decrement $\Delta i(s^*, t)$

next section.

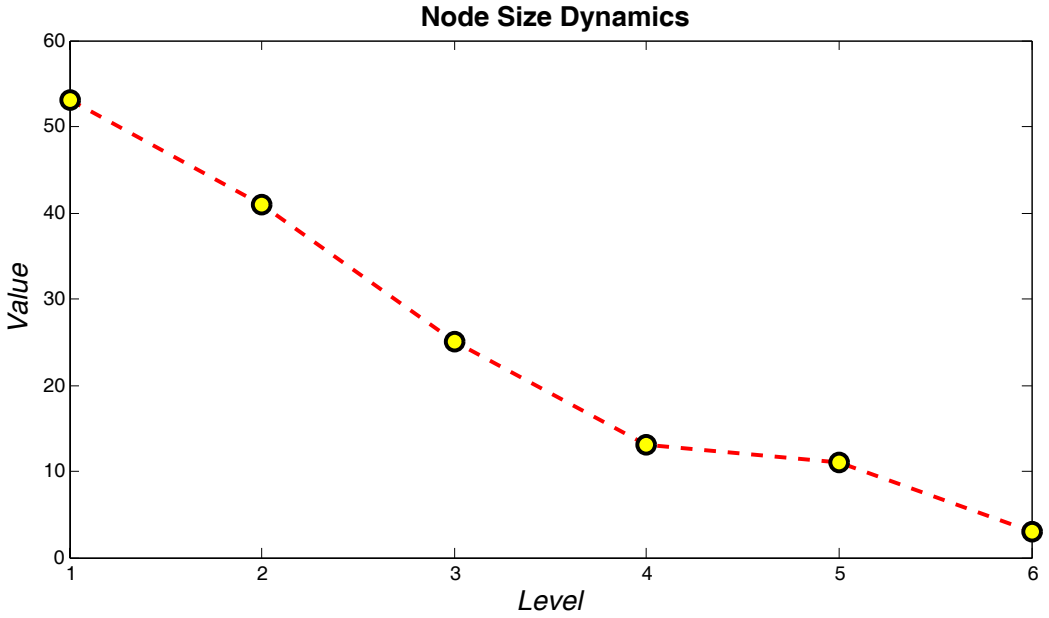

Figure 5.4: Node size values $n(t)$ computed for the dashed part of the tree on Figure 5.1 – from the root node to the terminal node

5.2 Cross-validation as a Method of Decision Tree Pruning

Two early stopping rules described in the previous section rely only on a given current configuration of a decision rule (i.e. the number of points in a node or current value of the impurity function decrement) and do not take into account any forecasting properties of such a rule. Cross-validation, instead, aims at comparing various configurations of a given classification rule based on the in-sample misclassification rate.

Cross-validation stands for a procedure that employs available data in the way that the bigger part of them is used as a *training set* and the rest – as a *test set*. Then the process is looped so that different parts of the data become the learning and training set so that at the end each data point is employed both as a member of the test and learning sets. The motivation behind such a recursive procedure is to extract the maximum amount of information from the learning sample especially in the situations of data scarceness.

The procedure is implemented in the following way. First, the learning sample is *randomly* divided into V parts: after that $(V-1)$ parts refer to the training set and one part – to the test set. Using the training set, a decision tree is constructed while the rest of the data (the test set) is employed to verify the tree quality since its actual class/response value is known from the learning sample (the division to the training and test sets is artificial). Note that to be able to assess the performance of the rule, the cost is the decreased number of observations in the training set (by $\frac{1}{V} \cdot 100\%$).

At the next step, the pool of data that was used as the test set becomes a part of the learning set, whereas another $\frac{1}{V}$-th part of the data becomes the *new* test set. The

looping ends when all data points were employed in such a way, i.e. when the maximum information from the data was extracted.

The aim of cross-validation is to compare the *quality of the tree* in various configurations, i.e. trees of different size. Define $L \backslash L_v$, $\forall v = 1, \ldots, V$ as the training set and L_v as the test set where L is the learning sample itself. For a given classification rule d_v based on the learning set $L \backslash L_v$, it is then possible to estimate the quality of the tree introducing the following misclassification rate:

$$E^1\left(d^{(v)}\right) = \frac{1}{n_v} \sum_{(l_n, j_n) \in L_v} \mathbf{I}\left(d^{(v)}(l_n) \neq j_n\right) \tag{5.3}$$

where l_n are test set observations with class tags j_n, $E^1\left(d^{(v)}\right)$ is a one-iteration estimate, and n_v is the size of the training set.

Since none of the observations from L_v were engaged during the construction of the decision rule $d^{(v)}$, it is then possible to define the *cross-validation measure of the tree quality* as

$$E^{CV}(d) = \frac{1}{V} \sum_{v=1}^{V} E^1\left(d^{(v)}\right) \tag{5.4}$$

or the averaged misclassification error (5.3) after all V steps of the loop.

The next important point is how to choose V. Although V is not an internal calibration parameter like $\bar{\beta}$, it may still be quite important because it is the key factor in the speed-precision tradeoff. It is worth noting that cross-validation can be *extremely slow* for big n (number of points in the learning sample) and V, hence an adequate balance is required. Usually, the value of V can be specified given the precise task formulation where the time constraint becomes extremely important. Imagine that an online classification system is required, e.g. for classifying different high-frequency stock exchange operations where the learning sample evolves practically every second. Then if one classification (including finding the optimal tree first because the learning sample changes very rapidly) takes, say, several minutes and the time constraint is only one second, cross-validation is obviously the wrong algorithm to apply. Nonetheless, for industrial settings either time constraints are usually not so extremely tough or the computing power is feasible to handle the required volume of computations because of the alternative costs (Breiman and Friedman, 1988). That is why the choice of the parameter V is mainly dependent on the feasible computing power.

Unfortunately, for small values of V, cross-validation estimates can be *unstable* because each iteration a cluster of data is selected *randomly* and the number of iterations itself is relatively small, thus the overall estimation result is somewhat random. Empirical simulations show that re-running the same procedure (with the same small value of $V \leq 10$) frequently results into noticeably different decision rules, especially when the underlying learning samples contain data with complex nonlinear links.

Since $n\left(1 - \frac{1}{V}\right) \to n$ for big $V \leq n$, the way to eliminate randomness in results is to increase V. In the limiting case, when $V = n$ and the single data point is employed as the test set (*leave-one-out cross-validation*), randomness obviously disappears, but only at the cost of the overwhelmingly increased amount of computations. In practical financial applications with high-dimensional data sets, the significant increase of V may

drastically increase the amount of computations, sometimes questioning the employment of this tree pruning method.

Cross-validation with $V = 10$ or $V = 25$ is a frequent default setup setting (Loh and Vanichsetakul, 1988b), and for many applications the achieved level of stability may be adequate.

But employing the cross-validation method itself for all possible tree configurations (i.e. the maximum tree and all smaller trees of various configurations) may also become infeasible (or costly) due to computation constraints. At this point the following question arises: is it possible to check not all subtrees of the maximum tree but only special *key subtrees*? With the results introduced in Breiman et al. (1987) it appears to be possible. The next section focuses on this challenge.

5.3 Cost-complexity Function and Cross-validation

The idea of the method that picks only 'significant' parts of the trees to be tested via cross-validation is to introduce some new measure that would be able to take into account *tree complexity*, i.e. its size, which can be estimated by the *number of terminal nodes*. Then the maximum tree gets a penalty for its big size, but on the other hand it will be able to make perfect in-sample predictions. Small trees get much lower penalty for their size, but their predicting abilities are naturally limited. The optimization procedure based on such a tradeoff criterion could determine the best decision tree size.

Define the *internal misclassification error* of an arbitrary observation at the node t as $e(t) = 1 - \max_j p(j \mid t)$, define also $E(t) = e(t)p(t)$. Then the *internal misclassification tree error* is $E(T) = \sum_{t \in \tilde{T}} E(t)$ where $\tilde{T}$ is a set of terminal nodes.

These estimates are called *internal* because they are based solely on the learning sample, which is opposite to, for instance, cross-validation that artificially introduces both learning and test sets. It may seem that using $E(T)$ as a tree quality measure is sufficient, but unfortunately that is not the case. Consider the situation when the maximum tree is built: $E(T_{MAX}) = 0$. In this case the tree should be concluded to have the best feasible configuration, but as it was discussed above (see Section 5.1), the maximum tree can represent optimal decision rules only in rare cases.

For any nested subtree $T \preceq T_{MAX}$, define the number of terminal nodes $\left|\tilde{T}\right|$ as the measure of its complexity. Then the following cost-complexity function could be used to optimize the decision tree size:

$$E_\alpha(T) = E(T) + \alpha \left|\tilde{T}\right| \tag{5.5}$$

where $\alpha \geq 0$ is a complexity parameter and $\alpha \left|\tilde{T}\right|$ is a cost component: the more complex is the tree (the higher is the number of terminal nodes) – the lower is the value of $E(T)$, but at the same time the value of the penalty $\alpha \left|\tilde{T}\right|$ is higher, and vice versa.

Although α can have infinite number of values, the number of subtrees of T_{MAX} resulting in the minimization of $E_\alpha(T)$ is *finite*. Hence pruning of T_{MAX} leads to the creation of a subtree sequence $T_1, T_2, T_3, \ldots$ with a decreasing number of terminal nodes.

Since the sequence is finite, if $T(\alpha)$ is an optimal subtree for some arbitrary α, then it will remain optimal until the complexity parameter is not changed to some value α' when $T(\alpha')$ becomes a new optimal subtree. The process loops when the complexity parameter value becomes α'' and so on.

The main question is if the optimal subtree $T \preceq T_{MAX}$ for a given α minimizing $E_\alpha(T)$ always *exists* and if it is *unique*. Moreover, for the reasons of the computational efficiency, one is interested if the sequence of optimal subtrees for different values of α is *nested*, i.e. $T_1 \succ T_2 \succ \ldots \succ \{t_0\}$ where t_0 is the root node (learning sample itself). When the sequence of subtrees is nested, the number of subtrees to check is reduced drastically.

In Breiman et al. (1987) it is shown that for $\forall \alpha \geq 0$ there exists an optimal tree $T(\alpha)$ in the sense that

1. $E_\alpha\left(T(\alpha)\right) = \min\limits_{T \preceq T_{MAX}} E_\alpha(T) = \min\limits_{T \preceq T_{MAX}} \left[E(T) + \alpha \left|\tilde{T}\right|\right]$
2. if $E_\alpha(T) = E_\alpha\left(T(\alpha)\right)$, then $T(\alpha) \preceq T$.

This result is then not only a proof of the existence, but also a proof of the uniqueness: consider another optimal subtree T' so that T and T' both minimize E_α and are *not nested*, then $T(\alpha)$ does not exist in accordance with the second condition.

The idea of the introduction of a cost-complexity function at this stage is to check only a *smaller subset* of different subtrees of T_{MAX} instead of all possible subtrees of T_{MAX} – only optimal subtrees for different values of α. The starting point is to define the first optimal subtree in the sequence so that $E(T_1) = E(T_{MAX})$ and the size of T_1 is minimum among other subtrees with the same cost level. To get T_1 out of T_{MAX} for each terminal node of T_{MAX}, it is necessary to verify the condition $E(t) = E(t_L) + E(t_R)$, and if it is fulfilled, the node t is pruned. The process is looped until no extra pruning is available – the resulting tree $T(0)$ becomes T_1.

Define the node t as an *ancestor* of t' and t' as *descendant* of t if there is a connected path down the tree leading from t to t'.

In the example on Figure 5.5, nodes t_4, t_5, t_8, t_9, t_{10}, and t_{11} are descendants of t_2 while nodes t_6 and t_7 are not descendants of t_2 although they are positioned lower. They are not descendants of t_2 because it is not possible to connect them with a path from t_2 without engaging t_1. Analogously, nodes t_4, t_2, and t_1 are ancestors of t_9, and t_3 is not an ancestor of t_9.

Define the *branch* T_t of the tree T as a subtree based on the node t and all its descendants. This branch can be considered as a separate tree.

Pruning a branch T_t from a tree T means deleting all descendant nodes of t. Denote the transformed tree as $T - T_t$. For the example on Figure 5.5, pruning the branch T_{t_2} will result in a new tree on Figure 5.7.

For any branch T_t, define an *internal misclassification estimate* as

$$E(T_t) = \sum_{t' \in \tilde{T}_t} E(t') \tag{5.6}$$

where $\tilde{T}_t$ is the set of terminal nodes of T_t. Hence, for an arbitrary node t of T_1 it is true that

$$E(t) > E(T_t). \tag{5.7}$$

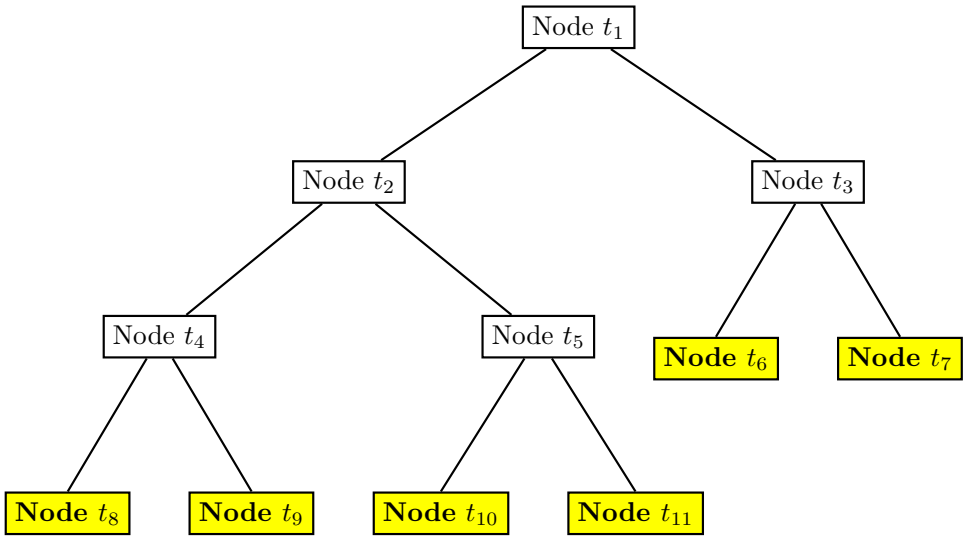

Figure 5.5: Decision tree hierarchy

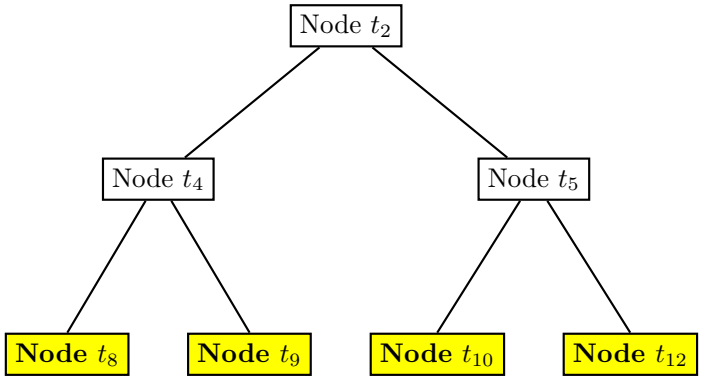

Figure 5.6: The branch T_{t_2} of the original tree T

Consider now the *cost-complexity misclassification estimate* for branches or single nodes. Define a single node estimator as

$$E\left(\{t\}\right) = E(t) + \alpha \tag{5.8}$$

where $\{t\}$ is a subtree consisting of the single node t. The branch cost-complexity misclassification estimate is then

$$E_\alpha(T_t) = E(T_t) + \alpha \left|\tilde{T}_t\right|. \tag{5.9}$$

When $E_\alpha(T_t) < E_\alpha\left(\{t\}\right)$, the branch T_t is preferred to the single node $\{t\}$ according to the cost-complexity misclassification estimate. But for some critical α, both values will

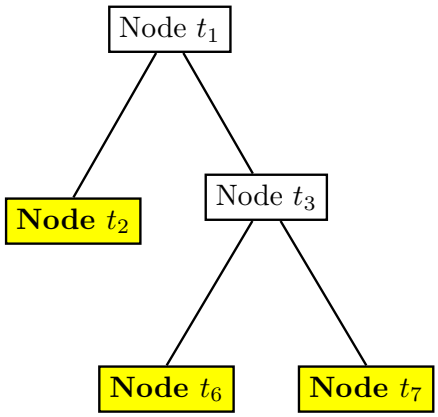

Figure 5.7: $T - T_{t_2}$: the pruned tree T

become equal – this critical value of α can be determined from the following inequality:

$$E_\alpha(T_t) < E_\alpha(\{t\}) \tag{5.10}$$

that is equivalent to

$$\alpha < \frac{E(t) - E(T_t)}{\left|\tilde{T}_t\right| - 1} \tag{5.11}$$

where $\alpha > 0$ because $E(t) > E(T_t)$.

To get the next member of the optimal subtree sequence, i.e. T_2 out of T_1, a special node called the *weak link* is determined. For this purpose a function $g_1(t)$, $t \in T_1$ is defined as

$$g_1(t) = \begin{cases} \frac{E(t)-E(T_t)}{\left|\tilde{T}_t\right|-1}, & t \notin \tilde{T}_1, \\ +\infty, & t \in \tilde{T}_1. \end{cases} \tag{5.12}$$

Then the node $\bar{t}_1$ is called a weak link in T_1 if

$$g_1(\bar{t}_1) = \min_{t \in T_1} g_1(t), \tag{5.13}$$

and the new value for α_2 is calculated as follows:

$$\alpha_2 = g_1(\bar{t}_1). \tag{5.14}$$

The new tree $T_2 \prec T_1$ in the sequence is defined by pruning the branch $T_{\bar{t}_1}$:

$$T_2 = T_1 - T_{\bar{t}_1}. \tag{5.15}$$

The process is looped until the root node $\{t_0\}$ – the final member of the sequence – is reached.

When there are multiple weak links detected, for instance $g_k(\bar{t}_k) = g_k(\bar{t}'_k)$, then both branches are pruned, i.e. $T_{k+1} = T_k - T_{\bar{t}_k} - T_{\bar{t}'_k}$.

In this way it is possible to get the sequence of optimal subtrees $T_{MAX} \succ T_1 \succ T_2 \succ$

$T_3 \succ \ldots \succ \{t_0\}$ for which one may prove that the sequence $\{\alpha_k\}$ is increasing, i.e. $\alpha_k < \alpha_{k+1}$, $k \geq 1$, and $\alpha_1 = 0$. For $k \geq 1$: $\alpha_k \leq \alpha < \alpha_{k+1}$ and $T(\alpha) = T(\alpha_k) = T_k$, see Breiman et al. (1987) for more details.

Practically, this suggests a method to implement the search algorithm. First, the maximum tree T_{MAX} is taken, then T_1 is found, after what the weak link $\bar{t}_1$ is detected and the branch $T_{\bar{t}_1}$ is pruned, α_2 is calculated, and the process is looped.

When the algorithm is applied to T_1, the number of pruned nodes is usually quite significant. A typical example is provided in Table 5.1.

Tree	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
$\left\|\tilde{T}_k\right\|$	71	63	58	40	34	19	10	9	7	6	5	2	1

Table 5.1: Typical pruning speed

When the trees become smaller, the difference in the number of terminal nodes also gets smaller.

Finally, it is worth mentioning that the sequence of optimally pruned subtrees is a subset of trees that might be constructed employing the direct method of the internal misclassification estimator minimization given a fixed number of terminal nodes. Consider an example of the tree $T(\alpha)$ with 7 terminal nodes. In this case, there is no other subtree T with 7 terminal nodes having lower $E(T)$. Otherwise

$$E_\alpha(T) = E(T) + 7\alpha < E_\alpha\left(T(\alpha)\right) = \min_{T \preceq T_{MAX}} E_\alpha(T),$$

which is impossible by definition.

Applying the method of V-fold cross-validation to the sequence $T_{MAX} \succ T_1 \succ T_2 \succ T_3 \succ \ldots \succ \{t_0\}$, an *optimal tree* is determined.

There is, however, one serious empirical drawback of this algorithm. The choice of a tree with the minimum value of $E^{CV}(T)$ is not always adequate since $E^{CV}(T)$ is not robust, i.e. there is a whole range of values of $E^{CV}(T)$ satisfying $E^{CV}(T) < E^{CV}_{MIN}(T) + \varepsilon$ for small $\varepsilon > 0$. Moreover, when $V < n$, a simple change of the random generator seed will very likely result in the changed values of $\left|\tilde{T}_k\right|$ minimizing $\hat{E}(T_K)$. Hence, a so called *one standard error* (1-SE) empirical rule is applied (Breiman et al., 1987). It states that if T_{k_0} is the tree minimizing $E^{CV}(T_{k_0})$ from the sequence $T_{MAX} \succ T_1 \succ T_2 \succ T_3 \succ \ldots \succ \{t_0\}$, then the value k_1 and the corresponding tree T_{k_1} are selected so that

$$\text{argmax}_{k_1}\, \hat{E}(T_{k_1}) \leq \hat{E}(T_{k_0}) + \sigma\left(\hat{E}(T_{k_0})\right) \tag{5.16}$$

where $\sigma(\cdot)$ denotes a sample estimate of the standard error and $\hat{E}(\cdot)$ – relevant sample estimates of misclassification rates.

The dashed line on Figure 5.8 shows the area where the values of $\hat{E}(T_k)$ only slightly differ from $\min_{\left|\tilde{T}_k\right|} \hat{E}(T_k)$. The left edge, which is roughly equivalent to 16 terminal nodes, shows the application of the one standard error rule: for robustness reasons, the optimal classification tree is chosen as the one containing 16 terminal nodes.

Theoretically, the use of the one standard error rule allows not only to achieve more robust results but also to get trees of lower complexity given the error comparable with

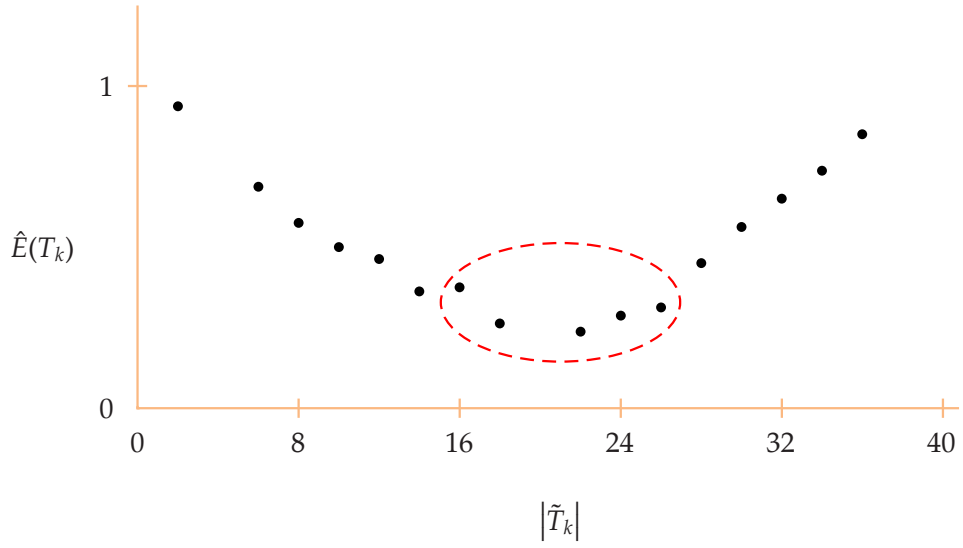

Figure 5.8: The example of a relationship between $\hat{E}(T_k)$ and number of terminal nodes. The red dashed line indicates the choice of trees having the comparable (within one standard error) empirical cost-complexity misclassification rate

$\min_{|\tilde{T}_k|} \hat{E}(T_k)$ – or the zero standard error rule (0-SE) as it is called alternatively.

Therefore, applying cross-validation to selected (in terms of cost-complexity) subtrees, one obtains a nonparametric procedure to get the optimal shape of the decision rule that drastically reduces the amount of necessary computations because not all of the subtrees – but only selected ones – are to be compared. However, neither $V = 10$ nor $V = 25$ for cross-validation guarantees the stability of the results (assuming the number of observations n is much greater than 25 points). Changing the random generator seed eventually influences the final results. This can be avoided at the cost of setting $V = n$ or repeating the procedure many times. However, even the increased number of computations does not solve the second problem – the minimization of the empirical cost-complexity function is not robust, therefore the empirical 1-SE rule is recommended. Sometimes, however, the 1-SE rule prunes the tree to such a degree that one of the predicted classes does not appear in the final rule, i.e. the tree is underfitted, and a more complex 0-SE tree has to be employed (Kim and Loh, 2001). But in this case, because the value of E^{CV}_{MIN} is not robust in the sense that several trees with different numbers of terminal nodes produce the comparable error, the critique to 0-SE trees applies, and one may conclude that the canonical cost-complexity approach is full of compromises.

However, this is a canonical method, and more recent and perhaps more advanced techniques of tree building and pruning are compared with this one in the literature. Section 6.2 provides a critical overview of these methods, especially in connection with the practical implementation of stock picking problem. As one will see below, more advanced and recent methods frequently show very similar results in terms of misclassification errors when the out-of-sample prediction is performed on reference data sets.

6 Critical Overview of Alternative Tree Building Techniques

6.1 Decision Tree Induction

6.1.1 FACT and QUEST

FACT is a method of construction of classification trees introduced in Loh and Vanichsetakul (1988a) that relies on the generalized discriminant analysis and has several key differences from the original CART method of Breiman et al. (1987). The most notorious one is that it uses linear combinations of input variables as split questions, so splits become multivariate. These linear combinations come as linear discriminant functions of the special form that are computed from selected principal components of the correlation matrix at each node. More specifically, a split is selected at the node t via the discriminant functions

$$d_i(y) = \hat{\mu}_j^\top \hat{\mathbf{\Sigma}}^{-1} y - \frac{1}{2}\hat{\mu}_j^\top \hat{\mathbf{\Sigma}}^{-1}\hat{\mu}_j + \ln\{p(j|t)\} \tag{6.1}$$

where y is the vector in the space of the larger (see below) principal components, $\hat{\mu}_j$ is the sample mean vector of the j-th class, and $\hat{\mathbf{\Sigma}}$ is the pooled estimate of the covariance matrix at the node. According to Loh and Vanichsetakul (1988a), only those principal components are taken whose eigenvalues exceed 0.05 (user-specified) times the largest eigenvalue.

For categorical variables, FACT can have multiple splits per node. Furthermore, there is no randomization in the procedure involved because cross-validation is not employed – a direct stopping rule is used instead. This rule is the following: splitting is stopped if the node apparent error rate does not decrease with splitting, or there is at most one class at the node with the sample size greater than a user-specified threshold value.

FACT is claimed to be computationally faster than canonical CART. The authors of the method conclude, however, that neither method dominates on the accuracy showing similar performance on the out-of-sample error estimates.

Breiman and Friedman (1988) do not regard this method, from a purely technical point of view, as a step forward in the tree induction evolution since it sacrifices traditional CART's pure nonparametric approach to be able to increase greatly the computational speed, which is rarely an issue for industrial settings. Moreover, Breiman and Friedman (1988) claim that the overwhelming majority of CART users prefer univariate splitting, and multiway splitting (three and more branches from the parent node) does not make as effective use of conditional information potentially present in the tree as does binary splitting. Finally, an early stopping rule as the tree optimization method is heavily criticized as not being able to produce reliable trees for various data sets – each stopping rule finally fails on some data set as does the one proposed in Loh and

Vanichsetakul (1988a).

From the practical point of view, FACT offers a competitive advantage in the speed, however two user-specified parameters have to be selected. And even when they are selected properly, the out-of-sample performance of FACT is generally on a par with CART according to the available empirical results.

Later, Loh and Shih (1997) presented an updated method called QUEST that shares similarities with the FACT method but yields binary splits and the final tree that can be selected by an early stopping rule or pruning. A splitting point is selected employing the concept of two superclasses to ensure binary splits when $J > 2$ – classes are grouped into two clusters before the discriminant analysis is employed (minimizing the within-cluster sum of squares). To accommodate unequal variances, QUEST uses a modified form of the quadratic discriminant analysis (QDA) on the two superclasses.

For the standard normal density function $\phi(x) = (2\pi)^{-1/2}\exp\left(-x^2/2\right)$ and the sample class means $\bar{x}^{(j)}$, $j = 1, 2$, the QDA splits the X-axis into three intervals $(-\infty, d_1)$, (d_1, d_2), and (d_2, ∞) where d_1 and d_2 are the roots of the equation

$$p\left(1|t\right) s_1^{-1} \phi \left\{ \frac{x - \bar{x}^{(1)}}{s_1} \right\} = p\left(2|t\right) s_2^{-1} \phi \left\{ \frac{x - \bar{x}^{(2)}}{s_2} \right\} \tag{6.2}$$

and s_j^2 denotes the sample class variance. In order to obtain a binary split, QUEST uses only one of the two roots as a splitting point: the one that is closer to the sample mean of each class.

While FACT uses the ANOVA F-statistic to choose the variable, QUEST goes further and introduces a procedure that offsets the variable selection bias via the Levene's F-statistic for unequal variances (Levene, 1960) – normally, even when all the variables are independent of each other and class variable, categorical variables are more likely to be chosen than ordered variables.

While the correction of the variable selection bias may be important for applications employing categorical variables, the empirical part of this work relies solely on ordered variables (see Chapter 8.3 and Table 8.2 for more details), therefore making this potential advantage of QUEST negligible. In Loh and Shih (1997) QUEST is claimed to be substantially faster than exhaustive search algorithms (like CART), although the size of its trees and classification accuracy are typically comparable, therefore providing no real benefits for the industrial application setting.

Furthermore, the equations (6.1) and (6.2) assume some specific forms of splits that may produce shorter trees in special separable cases and yield inferior results for more complex nonlinear structures where univariate splits are a more preferable choice according to Breiman and Friedman (1988).

6.1.2 ID3 and C4.5

ID3 and C4.5, an extension of ID3 that accounts for missing values, continuous attribute value ranges, and pruning of decision trees among some other features, is another tree induction method thoroughly described in Quinlan (1993).

C4.5 employs a tree induction measure different from that in standard CART – *information gain* and *gain ratio*. Let C denote the discrete class attribute that has values $C_1, C_2, \ldots, C_J$, let $\boldsymbol{X}$ be a fixed set of attributes $X = X_1, X_2, \ldots, X_p$, and S – the

training sample. Let $RF(C_j, S)$ denote the relative frequency of cases in S that belong to the class C_j. The information content of a message that identifies the class of a case in S is then

$$I(S) = -\sum_{j=1}^{J} RF(C_j, S) \log \{RF(C_j, S)\}. \tag{6.3}$$

After S is partitioned into subsets $S_1, S_2, \ldots, S_s$ by a question Q, the information gained is then

$$G(S, Q) = I(S) - \sum_{i=1}^{s} \frac{|S_i|}{|S|} I(S_i). \tag{6.4}$$

where $|\cdot|$ is the size of the set. The gain criterion chooses the question Q that maximizes $G(S, Q)$. However, the information gain measure tends to favor attributes with many values – for instance, $G(S, Q)$ is maximized by a question in which each S_i contains a single case. To solve this problem, the gain ratio criterion also takes into account the potential information from the partition itself:

$$P(S, Q) = -\sum_{i=1}^{s} \frac{|S_i|}{|S|} \log \left(\frac{|S_i|}{|S|} \right). \tag{6.5}$$

The gain ratio rule then chooses, from among the questions with at least average gain, the question Q that maximizes $G(S, Q)/P(S, Q)$.

Overfitting in C4.5 is avoided by means of computationally efficient pruning, which is implemented as a single bottom-up pass.

In Lim and Loh (2000) traditional CART and C4.5, among others, are tested on 32 data sets (16 original sets and 16 sets with added noise). It is concluded that CART (0-SE, 1-SE, $V = 10$) and C4.5 (information gain and gain ratio, default pruning settings) show no significant difference in performance although C4.5 tends to produce larger trees.

6.1.3 CHAID

CHAID, standing for *Chi-squared Automatic Interaction Detector*, is a successor of AID (Morgan and Sonquist, 1963b,a) and THAID (Morgan and Messenger, 1973). It can be used both for regression and classification problems and involves constructing many cross-tables and finding the most statistically significant proportions (via the chi-square test) that are employed to control the structure of a tree. When continuous variables are used, they are broken into a set of ranges similar to the appearance of categorical variables. CHAID is able to build non-binary classification trees, i.e. trees where more than two branches may go from a node. The original CHAID algorithm was introduced by Kass (1980) for nominal dependent variables.

CHAID proceeds in steps and the best partition for each predictor is found first. Then the best predictor is chosen based on mutual comparison, and the data are split according to this predictor. Similar to traditional CART, forthcoming data portions are split recursively.

Assuming that there are $d \geq 2$ categories of the dependent variable and a particular predictor has $c \geq 2$ categories, CHAID analyzes the data via a $c \times d$ contingency table. The given $c \times d$ table is then reduced to the most significant $j \times d$ table by combining

categories of the predictor. This is done in the following way. For each $j \times d$ table ($j = 2, 3, \ldots, c$), a statistic $T_j^{(i)}$ is calculated, which is the usual χ^2 for the i-th method of forming a $j \times d$ table. Then, if $T_j^* = \max_i T_j^{(i)}$ is the χ^2-statistic for the best $j \times d$ table, the most significant T_j^* is chosen. The distribution of T_j^* is discussed in Kass (1980). In practice, the Bonferroni adjusted p-values of the Pearson χ^2-test (for classification problems) and F-tests (for regression problems) are employed.

C4.5 yields a binary split if the selected variable is numerical. If it is categorical, the node is split into C subnodes where C is the number of categorical values. CHAID is similar to C4.5 but employs an additional step to merge some nodes (Kim and Loh, 2001).

6.1.4 Oblique Decision Trees

Breiman et al. (1987) introduced CART as a method employing univariate splits in the form of questions $X_i < x$? where X_i is the selected attribute and x is the question value. That class of decision trees may be called axis-parallel since the questions at each node are equivalent to axis-parallel hyperplanes in the attribute space. FACT, employing the LDA at each node, made a step forward to the type of trees that is called oblique.

Murthy et al. (1994), following the original idea of Breiman et al. (1987), propose the OC1 tree induction algorithm that creates splits of the following form:

$$\sum_{i=1}^{d} a_i X_i + a_{d+1} > 0 \tag{6.6}$$

where X_i are real-valued attributes and $a_1, \ldots, a_{d+1}$ are real-valued coefficients. Because node questions of this form are equivalent to hyperplanes at an oblique orientation to the axes, this class of decision trees is called oblique.

The first oblique decision tree algorithm to be proposed was CART with linear combinations (CART-LC) (Breiman et al., 1987) as a possible extension of axis-parallel CART. However, CART-LC has certain limitations (Murthy et al., 1994), the most notorious of which include:

- When finding coefficients $a_1, \ldots, a_{d+1}$, CART-LC frequently gets stuck in local minima, and there is no built-in mechanism to escape it.
- CART-LC sometimes makes adjustments that increase the impurity of a split, a feature that may be an attempt to escape local minima.
- There is no upper bound on the time spent at any node in the decision tree.

Another oblique decision tree algorithm, one that uses a very different approach from CART-LC, is the Linear Machine Decision Trees (LMDT) system (Utgoff and Brodley, 1991), which is a successor to the Perceptron Tree method (Utgoff, 1989). Each internal (i.e. non-terminal) node there is a Linear Machine – a set of J linear discriminant functions that are used collectively to assign an instance to one of the J classes. The training algorithm presents examples repeatedly at each node until the linear machine converges. Because convergence cannot be guaranteed, LMDT uses heuristics to determine when the node has stabilized.

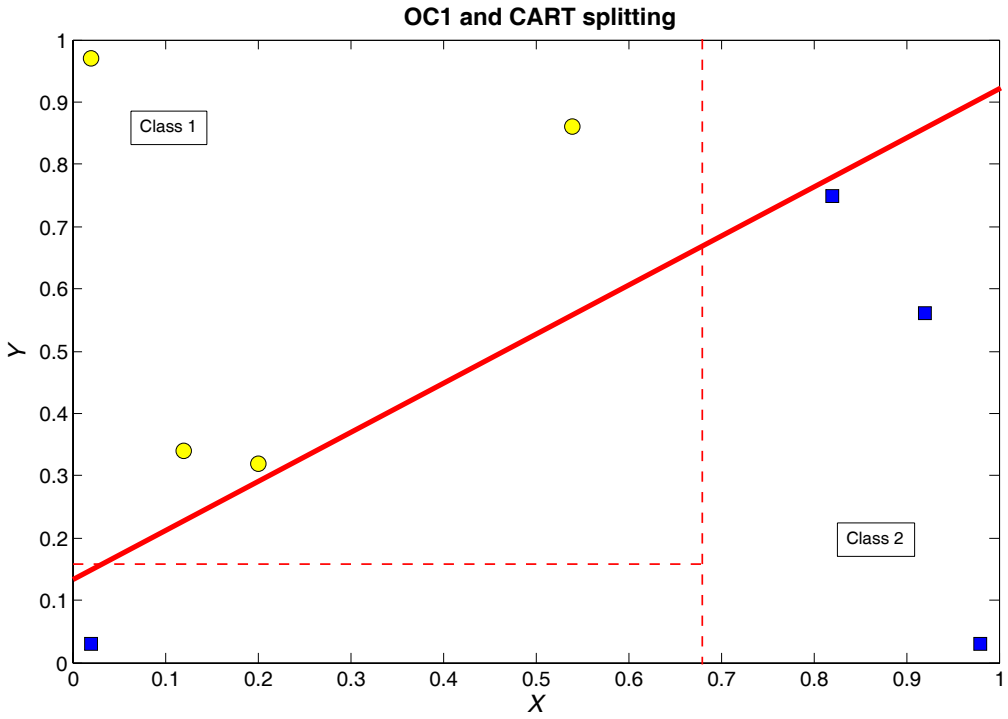

Figure 6.1: Oblique (OC1, solid line) and axis-parallel (CART, dashed lines) partitioning of a two-class data set

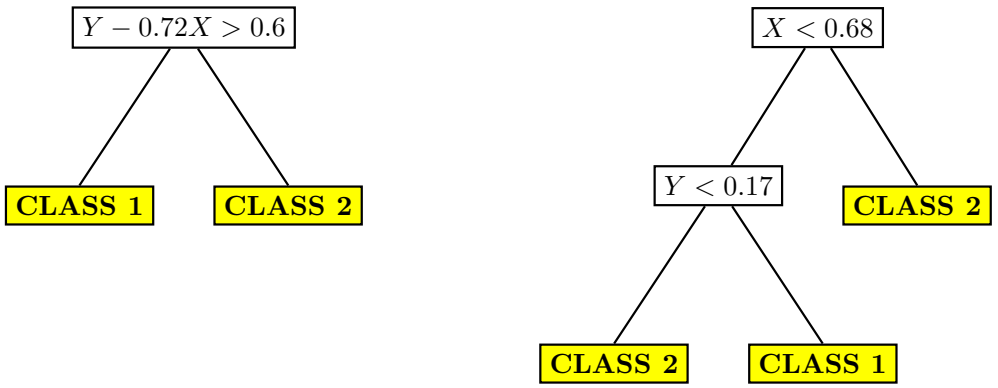

Figure 6.2: Respective OC1 and CART trees

Simulated Annealing of Decision Trees (SADT) is a method of oblique tree induction that uses randomization, which allows it to effectively avoid some local minima. However, that factor compromises on efficiency. The procedure runs much slower than either CART-LC, LMDT, or OC1.

Empirical experiments of Murthy et al. (1994) show that OC1 produces slightly better, but quite comparable, trees in terms of accuracy when OC1 is tested against CART-LC and C4.5. As expected, C4.5 trees are much more complex for each of the data set involved in the study. Slightly increased accuracy comes at the cost of computational

efficiency – OC1 uses deterministic hill climbing most of the time, however, it guarantees a worst-case running time that is only $\mathcal{O}(\log n)$ times greater than the worst-case time for inducing axis-parallel trees (i.e., $\mathcal{O}(pn^2 \log n)$ vs. $\mathcal{O}(pn^2)$), where n is the size of the learning sample and p is the number of features.

Figure 6.1 is an example (Murthy et al., 1994) of a two-class artificial learning sample that is employed for inducing OC1 and CART decision trees for comparison. Figure 6.2 presents both trees that filter observations of two classes in notoriously different ways. While the single split suffices to build the OC1 tree, CART has to use two splits because the first one ($X < 0.68$, vertical dashed line) leaves an observation of the second class in the pool of points belonging to the other class. OC1 produces a more compact tree, but from the equation $Y - 0.72X = 0.6$ it is impossible to assess which variable – X or Y – is more significant. The CART tree, in its turn, leaves no doubts here: the root node contains the question $X < 0.68$, and therefore it can be concluded that X is the variable that is the most significant, all other things being equal.

Therefore, being a slower and only a bit more precise method of decision tree induction with the potential lack of interpretability of produced decision rules, OC1 does no exhibit clear competitive advantages in comparison with CART when the stock picking task is considered.

6.1.5 Nonlinear Decision Trees

Starting with partitions of the feature space in the form of hyperrectangles that are parallel to the feature axes (axis-parallel decision trees like CART or C4.5), oblique decision trees introduce the first extra level of complexity – OC1 creates polygonal partitioning of the feature space. In contrast to CART and OC1, nonlinear decision trees (NLDT) produce even more complicated partitioning – in the form of curved hypersurfaces. Particularly, the form of nonlinear tree rules introduced in Ittner and Schlosser (1996a) is equivalent to hypersurfaces of the second degree.

NLDT is based on the combination of primitive features and the augmentation of the feature space before the tree generation. Because the space of possible new features is exponential, Ittner and Schlosser (1996a) consider only a special kind of the feature combination: all pairwise products and squares of p numerical primitive features. As a result, one obtains $\frac{p^2+3p}{2}$ features. These features are then considered as the axes of a new feature space, which are equivalent to terms of the equation of a hyperspace of the second degree. For a two-dimensional case, the equation takes the following form:

$$aX_1^2 + 2bX_1X_2 + cX_2^2 + 2dX_1 + 2eX_2 + f = 0, \tag{6.7}$$

and the augmented feature space contains variables $Y_1 = X_1$, $Y_2 = X_2$, $Y_3 = X_1X_2$, $Y_4 = X_1^2$, and $Y_5 = X_2^2$.

To produce splits in the form of the equation (6.7), the OC1 algorithm is applied to the augmented feature space. As one can, therefore, see, NLDT can be considered as a special case of OC1 – oblique partitioning in the higher-dimensional feature space corresponds to nonlinear partitioning of the primitive feature space.

Empirical tests performed by Ittner and Schlosser (1996a) show only minor (and almost indistinguishable) superiority of NLDT when its accuracy is compared with C4.5 and OC1 and several data sets like Fisher's iris are employed. Ittner and Schlosser

(1996a) claim a convincing demonstration of the usefulness of non-linear decision trees with respect to the accuracy – probably implying the results when an artificially created spiral data set was considered. There, NLDT performed significantly better – the accuracy of NLDT trees was almost up to two times better than of those produced by OC1. However, one should take into account a very specific pattern of the artificial data, and at this point of time it is worth addressing the response of the creators of CART when commenting on a similar setup – almost twofold superiority of FACT over CART given the spherical data set (Breiman and Friedman, 1988): *'It is true that the polar coordinate split does well in this particular case, because of its good match to the spherical symmetry built into the example. It is not likely that nonlinear decision boundaries will exhibit such nice symmetry in practice.'*

NLDT also leaves an open question to what extent the feature space should be augmented. Ittner and Schlosser (1996a) acknowledge this issue and attribute it to one of the directions of future research to be conducted.

Moreover, it is not clear whether the augmented feature space creates a negative impact on the stability of the produced decision rules because of the effect of correlation between primitive and augmented features such as X_1 and X_1^2.

NLDT takes a step forward in compactness of trees when compared to CART and OC1, which is understandable because of the ability of the method to incorporate more flexible filters due to the augmented feature space in a single node. As it was pointed out in Section 6.1.4, more compact trees have a disadvantage of lacking the interpretability. Nonlinear decision trees, therefore, suffer from this side-effect even to a greater extent than OC1.

One may, therefore, conclude that NLDT shows no significant competitive advantages over the aforementioned methods of tree induction when practical aspects of classification are concerned. As for the speed of NLDT, it is clearly on a par with OC1 or slightly less when the number of primitive features is large and the feature space is augmented significantly. Given the unstudied effects of the necessary degree of space augmentation and inevitable multicollinearity of the augmented feature space, NLDT can not be considered a technique practically outperforming CART.

6.1.6 Which Selection Measure for Tree Induction to Choose?

Traditional CART is an exhaustive search method producing binary decision trees. It employs pruning to find the optimal tree size via the 0-SE or 1-SE rules and is used as the reference for comparison with other methods.

FACT does not perform the exhaustive search – first, the variable with the largest F-statistic is selected and the linear discriminant analysis is applied to find the split point selection. This has a notorious disadvantage – the node is split into as many subnodes as there are classes. And if J is large, observations from the learning sample may be consumed too fast so that the tree may become too short to reveal hidden patterns of the data. Also note the difference with CART, where the variable and the splitting point are found simultaneously. From the technical point of view, FACT ignores the effect of unequal class variances.

QUEST purposefully produces only binary decision trees and accommodates unequal variances via the QDA on two superclasses. Similar to FACT, variable selection is performed via the Levene's F-statistic.

Because this study does not assume the presence of categorical variables (see Chapter 8.3 for data description), the difference between FACT and QUEST remains mostly in the way how the splitting point is found. However, as it was mentioned above, the misclassification rates of trees produced by FACT, QUEST, and CART do not differ significantly after standard pruning.

C4.5 relies on the greedy search that selects a split maximizing one of the heuristic splitting criteria – information gain or gain ratio. Empirical tests confirm significantly similar forecasting performance of C4.5 and CART, making CART, FACT, QUEST, and C4.5/ID3 practically equal from this point of view.

CHAID yields optimal splits based on the Pearson χ^2-statistic, and it is rather similar to C4.5, therefore being practically comparable with other reviewed methods.

Various forms of oblique decision trees differ mostly in their computational efficiency. CART-LC can get stuck in local minima, linear machines in LMDT are not guaranteed to converge, SADT is comparably slow, and only OC1 guarantees the bounded computation costs that are $\mathcal{O}(\log n)$ times greater than the worst-case time for inducing axis-parallel trees. However, the accuracy of oblique and axis-parallel trees is comparable. It is true that oblique trees are more compact, but their interpretability suffers a lot. Therefore, for the practical task of stock picking, oblique trees do not provide any significant advantages over univariate trees. Moreover, when the aforementioned methods of oblique tree induction other that OC1 are employed, the required computation time may increase manyfold.

The intuitively appealing idea of producing nonlinear splits and, therefore, potential ability of capturing more complex data links and creating trees with higher predictive potential is implemented in NLDT, a variant of OC1 with the artificially augmented feature space. However, NLDT does not offer practical advantages over axis-parallel splitting methods such as CART or C4.5 (except for some cases of artificial data sets where the internal structure represents some rare and geometrically sound structure like a spiral); instead, there are open questions concerning the necessary degree of space augmentation and multicollinearity of the primitive and augmented features.

More tree induction methods can certainly be developed and combined. For instance, Mingers (1989a) considers the empirical behavior of ID3 and several split selection measures. These measures include the *Quinlan's information measure*, the *G-statistic*, *using probabilities rather than the statistic*, *the Gini index*, *the gain-ratio measure*, and *the Marshall correction*. The results show that, after pruning, which was the standard cost-complexity variant in the study, the choice of a measure affects the size of a tree but not its accuracy, and therefore the classical procedure of CART suffices to describe and interpret the inner nonlinear links of data in the learning sample without the need for more complex and time-consuming computational techniques.

Perhaps the most important empirical result of Mingers (1989a) is that accuracy remains the same even when attributes are selected *randomly*. This fact suggests that the tree size – or pruning – is the most important element in tree prediction accuracy capabilities.

6.2 Alternative Pruning Methods

6.2.1 Pruning and Various Tree Induction Techniques

Evidence presented in Section 6.1 clearly shows that changes in the way axis-parallel splits are computed as well as the type of the tree – binary or multiway (three and more branches from the parent node) – do not have significant influence on the classification rules accuracy after standard pruning. More complex forms of trees – oblique and nonlinear – practically result only in the increased computation time and more compact trees. Such compact trees with, say, nonlinear splits may not only be more difficult to interpret but also have less flexibility in providing various subtrees for pruning purposes. Indeed, the same data structure may be described by an axis-parallel maximum tree containing, say, 12 terminal nodes or – alternatively – by a nonlinear maximum tree containing only two terminal nodes when the observations are separable by a curved hypersurface of some higher order, and if an axis-parallel tree with 12 terminal nodes has enough flexibility in terms of selecting various subtrees, a nonlinear tree with only two terminal nodes has no options to alter the classification rule in any way. Chances are high that such a rule is overfitted.

Because of these considerations, only axis-parallel and OC1 trees will be considered below.

Breiman et al. (1987) point out that pruning is the most important part of the tree building. Mingers (1989b) provides the following estimate: while achievable accuracy differs between domains, pruning improved accuracy by 20% to 25% for most domains. These differences were also found to be statistically significant. One should, however, note that not all data sets (although their majority in practice) benefit from pruning, see Esposito et al. (1997) for more details.

While pruning can be considered one of the crucial elements of tree induction, there have been several attempts to introduce pruning algorithms that are alternative to traditional cost-complexity pruning (the canonical method is thoroughly described in Chapter 5). The following sections introduce the major concepts behind these methods and evidence on their performance in practice based on the works of Mingers (1989b) and Esposito et al. (1997).

6.2.2 Critical Value Pruning

Unlike cost-complexity pruning, which requires the maximum tree to be built first, *critical value pruning* (Mingers, 1987) relies on estimating the importance of a particular node from online calculations performed during the tree creation stage, so the maximum tree is very likely not to be reached at all. As it can be seen from Sections 4.2 and 6.1, particular splits are determined as a result of the optimization of a given goodness of slit measure, which indicates how well the chosen filter separates the observations between classes in a given node. Critical value pruning keeps the current branch and does not prune the nodes when the threshold (critical) value is reached. Larger critical values lead to more compact trees.

Because the choice of the critical value is not obvious, the other variant of the procedure creates several decision rules for multiple critical values so that only one of them – the optimal one – can later be selected, which resembles to a great extent the procedure of cost-complexity pruning where various values of α in the equation (5.5) create

a sequence of subtrees containing the optimal rule.

The denser is the grid for critical values – the higher is the chance not to miss the tree that may supersede the optimal tree when the grid contains fewer critical values. Unlike cost-complexity pruning that automatically detects the number of various critical values α sufficing to obtain the optimal rule, critical value pruning does not provide any hints about the choice of the parameter or the necessary grid size – the particular critical value depends on the tree induction measure used and may depend on the data set, too.

Does critical value pruning offer some practical advantages when compared to cost-complexity pruning? It may do so under two conditions: first, the cost-complexity subtree sequence does not include some tree that may exhibit better out-of-sample performance than the optimal one derived from the minimization of (5.5); second, the grid of critical values of critical value pruning must be so dense to contain the virtual tree, which actually may not exist at all in the end, that produces better accuracy than the optimal cost-complexity tree. Meeting these conditions is very unlikely, the difference between critical value pruned and cost-complexity optimal trees in terms of accuracy may be negligible, and the computation time (due to the denser grid of critical values) may be overwhelming.

6.2.3 Minimum-Error Pruning

The aim of bottom-up *minimum-error pruning* developed by Niblett and Bratko (1987) is to find a single tree that minimizes the expected error rate on an independent data set. The method takes T_{MAX} and compares the expected error from pruning with expected error without pruning: the expected error of each internal node (or the so called static error) is matched against the weighted sum of the expected error rates (dynamic error) of the given node's children. This is implemented via the so called *m-probability estimate* – the expected probability that an observation reaching the node t belongs to the j-th class is computed as following:

$$p(j|t) = \frac{n_t(j) + \pi_j m}{n_t + m} \tag{6.8}$$

where π_j is the a priori probability of the class j and m is a parameter that generally controls the degree of pruning by influencing the link between the a priori and a posteriori probability π_j and $p(j|t)$. Niblett and Bratko (1987) assume m to be equal for all classes for simplicity.

The expected error rate $E(t)$ for a node t is then computed as following:

$$E(t) = \min_j [1 - p(j|t)] = \min_j \left[\frac{n_t - n_t(j) + (1 - \pi_j)m}{n_t + m} \right]. \tag{6.9}$$

Niblett and Bratko (1987) assume $m = J$ and $\pi_j = \frac{1}{J}$, so the equation (6.9) transforms to:

$$E(t) = \min_j \left[\frac{n_t - n_t(j) + J - 1}{n_t + J} \right]. \tag{6.10}$$

The dynamic error for branches is calculated from (6.10) with weights defined in (6.8).

According to Mingers (1989b), there are several problems with minimum-error pruning. The method assumes equally likely classes in the learning sample, which does not

happen often in practice, however the effect of this deviation is not clear. Secondly, Mingers (1989b) criticizes the method because it exhibits unstable results when the number of classes in the learning sample is changed. Greater values of m imply the more severe degree of pruning, however Esposito et al. (1997) acknowledge that this is not always the case, and in practice the non-monotonicity property results in the drastically increased volume of computations – for increasing values of m, the pruning process must always start from T_{MAX}.

6.2.4 Reduced-Error Pruning

Reduced-Error Pruning (Quinlan, 1987) is probably the simplest conceptual tree pruning technique. It starts with T_{MAX} and in the end produces a series of pruned trees. At each internal tree node t of T_{MAX}, it compares the number of classification errors of T and $T - T_t$ where T is a current tree configuration. If the simplified tree has better performance than that where the branch T_t is kept, the procedure recommends to prune T_t. T_t can be pruned only if it contains no subtree that results in a lower error rate than that for T_t itself. The pruning algorithm loops on simplified trees until further pruning leads to the increased misclassification rate.

Esposito et al. (1997) formally prove that reduces-error pruning finds the smallest version of the most accurate subtree with respect to the pruning set. The same study acknowledges the obvious positive property of the method – its linear computational complexity (each node is visited only once to evaluate the opportunity of pruning it). It is concluded, however, that overall the method has a bias towards overpruning.

6.2.5 Pessimistic Error and Error-based Pruning

Pessimistic Error Pruning is another pruning method introduced in Quinlan (1987). Similar to minimum-error pruning, it introduces a correction to one of the key measures involved in pruning calculations. More formally, Quinlan (1987) is concerned with the fact that the same training set is used for both growing and pruning a tree, and the error rate on the training set is likely to be optimistically biased and should not be used to choose the best pruned tree that is likely to come overly large. A *'more realistic error rate'* is provided.

If e_t is the number of examples from the learning sample that are misclassified at the node t, the initial estimate of the misclassification rate at the node t is

$$r_t = \frac{e_t}{n_t}. \tag{6.11}$$

Quinlan (1987) introduces the continuity correction for the binomial distribution, and the rate with the continuity correction takes the following form:

$$r'_t = \frac{e_t + 1/2}{n_t}. \tag{6.12}$$

Quinlan (1987) points out that although for (6.12) the situation when a subtree always makes fewer errors than the corresponding node no longer holds because the corrected estimates depend on the number of leaves and not just on the number of errors, it is likely that even this corrected estimate of the number of misclassifications made by the

subtree will be optimistic. Therefore, pruning of the subtree is suggested if its corrected misclassification rate exceeds the node counterpart by more than one standard error.

Esposito et al. (1997) point out that the introduction of the continuity correction in the estimation of the error rate has no theoretical justification because in statistics it is used to approximate a binomial distribution with a normal one, but it was never applied to correct overoptimistic estimates of error rates. It is concluded that the constant 1/2 from the equation (6.12) is suitable in some problems but not others. Mingers (1989b), however, claims that although the method is heuristic and the continuity correction is incidental, the method is successful, does not require a test data set, and it is very quick because it only has to make one pass and only looks at each node once.

One of the aims of *error-based pruning* (Quinlan, 1993) (the pruning method in C4.5) was to introduce even a more pessimistic estimate of the expected error rate. Apart from this novelty, error-based pruning combines pruning and *grafting* – substituting a branch of the tree with another branch of the same tree. The sum of the predicted error rates of all the leaves in a branch T_t is considered to be an estimate of the error rate of the branch itself. The predicted error rate is compared for the node t, branch T_t, and the largest subbranch $T_{t'}$ rooted in a child t' of t, and the conclusion whether to prune T_t, to graft $T_{t'}$ in the place of t, or to keep the original T_t is made.

Potential grafting is one of the distinct advantages of the method, however the present implementation may lead to some undesirable effects that can be avoided, though, quite easily, see Esposito et al. (1997) for more details. Esposito et al. (1997) also question the assumption of the method that treats the training examples covered by a node t of T_{MAX} as a statistical sample and the assumption that errors in the sample have a binomial distribution. Another interesting conclusion of the study is the fact that although error-based pruning employs a far more pessimistic estimate of errors than that adopted in pessimistic error pruning, experimental results lead to the very opposite conclusion.

6.2.6 MDL- and MML-based Pruning

Minimum Description Length (MDL) is one of the recent methods that solves the problem of model selection in a way that is different to techniques presented above. MDL assumes that fundamental data links (or regularities) can be employed to compress the data – to describe them using fewer symbols (or bits) than the number of bits when the data are described literally. Different tree configurations are equivalent to different ways of learning and, therefore, compressing the original information. MDL employs the principle of Occam's Razor and chooses such a model that is the best tradeoff between its complexity and goodness of fit.

A tree can then be regarded as a means for encoding classes of samples in the training database given a set of predictor attributes (Blazewicz et al., 2003). The best tree is the one that uses the least number of bits. To be able to compare different encoding schemes, encoding costs should be defined first.

Encoding the type of a node (terminal or internal) requires 1 bit of information. The cost of encoding an internal node t consists of the cost of the splitting attribute (variable number for continuous data) and the cost of the splitting predicate (question value for continuous data). Given p predictor attributes, $\log p$ bits are required to encode the splitting predicate. If an attribute has $v - 1$ splitting points (i.e. the attribute has v different values in the learning sample), encoding a split point requires $\log(v - 1)$ bits

for continuous data and $\log(2^v - 2)$ bits for categorical data.

The cost of encoding a tree is the sum of costs of encoding each node of the tree. If the cost of a splitting criterion at a node t is $C_{split}(t)$, then the cost of encoding an internal node is

$$C_{internal}(t) = C_{split}(t) + 1. \tag{6.13}$$

The cost of encoding a terminal node includes the cost of encoding class tags of all observations assigned to this node. If a terminal node t contains n observations belonging to m classes C_i, $i = 1, \ldots, m$, and s_i is the number of points belonging to the class C_i, then the amount of information necessary to clarify a given point at the node t is

$$E = -n \sum_{i=1}^{m} \frac{s_i}{n} \log \frac{s_i}{n}, \tag{6.14}$$

and the cost of encoding a terminal node t with n points is

$$C_{terminal}(t) = nE + 1. \tag{6.15}$$

Given these encoding principles, the optimal decision tree is found via bottom-up pruning according to a recursive procedure. Assuming that an internal node t has child nodes t_L and t_R, let $minC_t$ denote the cost of encoding the minimum cost subtree rooted at t. Child nodes t_L and t_R are pruned and t is transformed into a terminal node if the cost of encoding the class labels of observations at t is lower than or equal to the cost of encoding the subtree rooted at t:

$$C_{terminal}(t) \leq C_{internal}(t) + 1 + minC_{t_L} + minC_{t_R}. \tag{6.16}$$

Mehta et al. (1995) perform empirical comparison of MDL-based pruning with cost-complexity, pessimistic error, and C4.5 pruning and conclude that MDL pruning produces accuracy comparably or slightly better than that achieved with other pruning algorithms. MDL-pruned trees are more compact than those produced by pessimistic error pruning or C4.5, however not as compact as in the case of the cost-complexity algorithm. On the other hand, more compact cost-complexity trees take longer time to compute.

MDL shares some ideas with the *Minimum Message Length*(MML) principle that predates MDL by 10 years. As in MDL, MML chooses the structure that minimizes the length of the encoded data. However, the codes employed by MML are quite different from those in MDL. Nevertheless, in practice it leads to similar results, see Grünwald and Rissanen (2007) for more details.

6.2.7 Pruning Using Multiple Performance Measures

Pruning methods described above choose the best decision tree on the basis of its validation accuracy, and if multiple trees yield the same validation error rate, according to the Occam's Razor principle, the tree with the smallest number of leaves is selected (note that the issue of the tree size is addressed only in the case of a tie). While the validation error is certainly an important indicator of the produced decision rule forecasting capabilities, sometimes the sole use of this criterion – minimization of the error rate

– may be not optimal. Osei-Bryson (2007) provides the following example: a subtree with a validation data set accuracy rate of 0.959 and 29 leaves would be selected over a subtree with a validation data set accuracy rate of 0.958 and 5 leaves. While the difference between the validation accuracy of both trees is indistinguishable, for applications critical to the complexity of produced rules, the proposed choice may be not optimal. The motivation for pruning that takes into account multiple measures (Osei-Bryson, 2004, 2007) rises from similar examples when other decision rule characteristics such as stability or simplicity may be of particular importance for end-users.

Osei-Bryson (2004) provides a through review of additional measures of this kind that include:

- *Stability*
  This performance criterion should ensure that the accuracy rate does not vary significantly when the decision tree is applied to different data sets. If ACC_V and ACC_T are accuracy rates for validation and training respectively, the measure is defined as following:

$$Stab = \min\left\{\frac{ACC_T}{ACC_V}, \frac{ACC_V}{ACC_T}\right\}. \tag{6.17}$$

  $Stab \in (0; 1]$ and higher values of $Stab$ indicate higher stability of a decision rule. A finer version of the measure that focuses on the relative class frequencies of each leaf based on the validation and training data sets can be defined, too, see Osei-Bryson (2004) for more details.

- *Simplicity*
  When a decision tree is employed both as a descriptive and predictive tool, the high level of interpretability may be important, and simplicity is usually defined as a function of number of leaves in the decision tree and the rule length.

  – Based on the number of leaves

$$SIMPL_{Leaf} = f_{Leaf}\left(\left|\tilde{T}\right|\right) \tag{6.18}$$

    where $f_{Leaf}(\cdot)$ is an increasing utility function such that $SIMPL_{Leaf} \in (0; 1]$ and $\left|\tilde{T}\right|$ is the number of terminal nodes of the tree T. Higher values of $SIMPL_{Leaf}$ indicate higher simplicity.

  – Based on the average chain length
    Any decision tree can be regarded as a combination of 'if-then' rules. Given a rule, its length is defined as the number of predictor variables involved in this rule. Let φ_{V_t} be the proportion of the validation data set cases that are associated with the node t and L_t be the rule length for $t \in T$ where T is the analyzed tree. The mean rule length of the decision tree T can then be defined as

$$L_{Mean} = \sum_t \varphi_{V_t} L_t, \tag{6.19}$$

which is a weighted sum of the lengths of each rule. The corresponding simplicity measure is defined as

$$SIMPL_{Rule} = f_{Rule}(L_{Mean}) \tag{6.20}$$

where $f_{Rule}(\cdot)$ is a non-increasing function such that $SIMPL_{Rule} \in (0; 1]$ and higher values of $SIMPL_{Rule}$ indicate higher simplicity. Note that the lower bound for the simplicity rule is also likely to be active when overpruned trees are to be avoided.

- *Discriminatory power*
  Higher discriminatory power is defined as lower ambiguity of the class to which an observation is to be assigned, i.e. the higher posterior probability of a class. Define τ as the cut-off value for the posterior probability such that the user would be comfortable with the decision associated with this leaf only. Let ρ_{T_t} be the posterior probability of the decision event, and let $\Psi(t) = 1$ if $\rho_{T_t} \geq \tau$; and $\Psi(t) = 0$ if $\rho_{T_t} < \tau$. The accuracy measure based on discriminatory power can then be defined as
  $$DSCPWR = \sum_t \varphi_{V_t} \Psi(t) \tag{6.21}$$
  where $DSCPWR \in [0; 1]$, with higher values of DSCPWR indicating higher discriminatory power.

When the end-user selects several performance measures to compare various trees, technically the task becomes a multiple criteria decision-making problem. Although various formal techniques have been proposed including the weighing model and outranking methods, Osei-Bryson (2004) employs the weighing model because of its popularity, relative simplicity, and intuitive appeal. According to this model, the composite score of a tree is computed as a weighted sum of its performance based on the individual measures. Various approaches to estimate these weights are available. For instance, commercial expert choice software systems provide tools for elicitation of pairwise comparison of data from evaluators after what the associated weight vectors are automatically generated.

6.2.8 Which Pruning Method to Choose?

Chapter 5 presented the standard cost-complexity pruning methodology, the current chapter described the popular alternatives that employ either similar or completely different techniques when deriving the optimal tree size.

Critical value pruning introduces a user-defined parameter to gain speed, however wrong parameter selection may result in severe deterioration of a classification rule. The way to determine the 'correct' critical value is not provided, therefore one is encouraged to employ a grid of critical values and compare the resulting trees (for instance, via the cross-validation procedure). In Section 6.2.2 two conditions when critical value pruning may overperform cost-complexity pruning are stated, however one has to conclude that these conditions are rather unlikely to happen, the performance difference (if any) is very likely to be negligible, and the computational expenses may be disproportionate (due to the very dense grid of the critical values). Hence, critical value pruning can not be considered as a more competitive pruning technique.

It may be much more difficult to assess the capabilities of the other aforementioned pruning methods in advance due to their sometimes controversial assumptions (pessimistic error pruning), different architecture (introduction of potential grafting in error-based pruning), top-down (critical value pruning) vs. bottom-up approaches, etc. Empirical comparison can be considered one of the natural ways to answer this question.

There are at least two major studies, which have already been mentioned above, – Mingers (1989b) and Esposito et al. (1997) – that independently performed such a comparison on various data sets using slightly different methodologies. Mingers (1989b) concludes that there are significant differences between the methods. Minimum-error pruning produces noticeably different levels of pruning even on essentially the same set of data and is very sensitive to the number of classes in the data. As a result, it is suggested to be the least accurate method. Pessimistic error pruning gave bad results on some of the employed data sets and is recommended to be treated with caution. Critical value, traditional cost-complexity, and reduced-error pruning methods performed well producing consistently low error rates over all the data sets. However, critical value pruning requires the specification of initial and incremental values and can be slow if these are not chosen appropriately.

The second extremely important conclusion of Mingers (1989b) is that there is no evidence of the interaction between the type of the measure used in tree creation and pruning method, which means that empirical results on various pruning methods performance are very likely to hold even when other tree induction mechanisms are employed.

Unlike Mingers (1989b), Esposito et al. (1997) do not rely on the Analysis of Variance (ANOVA) to detect statistically significant differences between pruning methods. The authors motivate this decision by the fact that the ANOVA test is based on the assumption that the standard deviation is constant for all the experiments, whereas this is not so – the algorithms are compared on different data sets, each of which has its own standard deviation. A two-tailed paired t-test for each experiment is employed instead.

Esposito et al. (1997) claim that reduced error pruning and 1-SE cost-complexity pruning (see also Section 5.3) is prone to overpruning (underfitting), and minimum error pruning, critical value pruning, and error-based pruning tend to underprune.

From the practical point of view, it is concluded that there is no indication that methods exploiting an independent pruning set definitely perform better than the others. According to Esposito et al. (1997), the discrepancy with results of Mingers (1989b) should be attributed to the different design of the experiments and the fact that Mingers (1989b) employed four selection measures vs. only one in Esposito et al. (1997).

One should also keep in mind the significance testing methodology of Mingers (1989b) that can follow unrealistic assumptions (as mentioned above). Although visual inspection of the misclassification rates produced by various pruning techniques on five data sets with four tree induction measures in Mingers (1989b) reveals some performance difference, this difference does not seem to be *practically* significant in the majority of cases (the absolute difference between misclassification rates is frequently only a couple of percentage points).

Although MDL pruning is based on completely different assumptions as compared to other presented pruning methods, its empirically assessed accuracy performance is very close to that yielded by cost-complexity pruning and other methods. This is a fast and intuitively appealing method, but it tends to produce larger trees, which are more difficult to interpret.

Multiple performance measures pruning presented in Osei-Bryson (2004, 2007) is an attempt to meet potentially more sophisticated needs of the end-user when the accuracy measure does not suffice to describe fully the 'quality' of a decision rule. Multiple performance measures can be employed to construct a score index of a tree, and by comparing scores of different subtrees, pruning can be carried out. Many elements in such pruning are user-defined (for instance, utility functions or various threshold values), the way how the performance of a tree is assessed is not limited to the mere accuracy rate, therefore direct comparison of this pruning method with other aforementioned techniques is not feasible. However, conceptually, the indication of the multiple criteria assessment of a decision rule is already very important. As it will be seen from Chapter 7, some parallels between multiple performance measures pruning and the novel pruning methodology presented in this work can be drawn.

One can therefore conclude that different pruning methods presented in this section show very similar performance in terms of accuracy but differ significantly when the tree size and execution time are taken into account. Based on the experimental evaluations of Mingers (1989b) and other empirical studies, Murthy et al. (1994), for instance, chose cost-complexity pruning as the default pruning algorithm of OC1 tree building.

6.3 Ensemble Methods

6.3.1 Ensemble Methods in Machine Learning

A class of learning algorithms that constructs a set of classifiers and classifies new data by taking a weighted or unweighted vote of their predictions is called *ensemble methods*. Ensemble methods are not a prerogative of decision trees, instead they are widely applied to various techniques of machine learning. However, their conjunction with decision tree methodology emerged several recognized classification methods including so called *random forests*, a technique to be discussed in Section 6.3.3.

Dietterich (2000a) provides an excellent overview of various ensemble methods and points out that a necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse. An accurate classifier is defined as the one that has an error rate of better than random guessing on new $\boldsymbol{X}$ values. Two classifiers are called diverse if they make different errors on new data points.

While it is clear that various decision tree classifiers are very likely to be accurate and diverse in the aforementioned sense, it is questionable how much new ensemble classifiers would increase the accuracy rate and under what particular conditions that benefits would be tangible.

At the same time, the potential increase of the accuracy rate is very likely to cause the significant increase of the required computation time (depending on how many instances of initial classifiers are pooled for voting). While some of the algorithms have the potential to offset at least partially this drawback (see Section 6.3.3 about random forests for such an example), other decision rule quality measures such as interpretability and rule size may deteriorate to an unacceptable level.

The next sections provide a critical overview of several popular ensemble methods applied to decision trees and their prospects of being employed as an effective solution to the stock selection problem.

6.3.2 Bagging – Bootstrap Aggregating

Bagging predictors are one of the methods for generating multiple versions of a predictor and using these to get an aggregated predictor introduced in Breiman (1996a). The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class.

Let L be the learning sample consisting of n independent observations and $d(\boldsymbol{X}, L)$ – the produced decision rule. Suppose that a sequence of learning sets $\{L_k\}$, each consisting of n independent observations from the same underlying distribution, is available. Then an aggregated decision rule $d_A(\cdot)$ for numerical outcomes can be obtained simply as

$$d_A(\boldsymbol{X}) = \mathbb{E}_L d(\boldsymbol{X}, L) \tag{6.22}$$

where $\mathbb{E}_L$ denotes the expectation over L.

However, in practice several learning samples are usually not available, and Breiman (1996a) proposes to imitate the process in (6.22) by taking repeated bootstrap samples $\left\{L^{(B)}\right\}$ from L:

$$d_B(\boldsymbol{X}) = \mathbb{E}_L d\left(\boldsymbol{X}, L^{(B)}\right). \tag{6.23}$$

For classification trees, where Y is a vector of class tags, $d\left(\boldsymbol{X}, L^{(B)}\right)$ vote to form $d_B(\boldsymbol{X})$. This procedure is called *bootstrap aggregating* or *bagging* in Breiman (1996a).

The $\left\{L^{(B)}\right\}$ form replicated data sets, each consisting of n points, drawn from L at random but with replacement. Therefore, each observation (x, y) may appear several times or not at all in any particular $\left\{L^{(B)}\right\}$.

If changes in L produce only small changes in d, then d_B will be close to d. Generally, Breiman (1996c) concludes that decision trees and neural networks are examples of unstable procedures, and therefore bagging should lead to better accuracy rates of aggregated rules. Breiman (1996a) provides solid statistical reasoning why bagging actually works and why it is driven by unstable classifiers for both types of Y – numerical and categorical.

Empirical comparisons of bagging and single tree classifiers (pruned, 10-fold cross-validated) on 7 data sets in Breiman (1996a) show high *relevant* efficiency of bagging, however in absolute terms this difference may not be so significant, especially for practical applications. For instance, Breiman (1996a) reports a stunning 47% decrease of the misclassification error rate on the *heart* data set, however in absolute terms the rate of correctly classified observations increased from 90% to 94.7%. This 4.7% absolute increase of the classification rate is certainly important, however the sole figure of 47% can be impressive and misleading simultaneously, especially if one takes into an account that a 4.7% absolute increase (and the biggest among seven employed data sets relative 47% increase) resulted in the fiftyfold increase of the computation time because the number of bootstrap samples was equal to 50 in this experiment. At the same time, the biggest absolute attained difference is 9.6% (and a 33% relative decrease of the misclassification rate) while the mean absolute decrease of the misclassification rate for 7 data sets is 4.9%. According to Breiman (1996a), 50 bootstrap replicates for classification tasks are not necessary or sufficient, but simply these numbers *'seemed reasonable'*; the *waveform* data set example showed that with 25 bootstrap replicates the accuracy rate

has almost no potential for improvement. That may be not true for other data sets, though, and no relevant figures for other 6 data sets are provided in the study.

Clearly, the efficiency of bagging tree classifiers depends on the particular learning sample. If the accuracy rate of a classification procedure is crucial for the end-user, then even a fiftyfold increase of computational expenses may justify an average 4.9% (for the reported 7 data sets) absolute increase of accuracy. However, the structure of the new ensemble rules becomes invisible with no possibilities to visualize and interpret the results as in the case when single classification trees are employed.

6.3.3 Random Forests

Random forests are an ensemble method of decision tree building introduced in Breiman (2001) that combines bagging and randomization of individual splits of a special kind. Moreover, it operates only with maximum trees and produces a final decision rule that does not overfit (under some technical conditions, see below).

Dietterich (2000b) examines the practical benefits of building a specific type of an ensemble rule where splits in single decision trees are determined at random from a set of several opportunities. More specifically, let K be the number of best splits. When generating the k-th decision tree, let $\mathbf{\Theta}_k$ be a random matrix independent of past random matrices $\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_{k-1}$ such that a matrix $\mathbf{\Theta}_k$ consists of a number of independent random integers between 1 and K, which are employed sequentially to indicate the randomly chosen split. The nature and dimensions of $\mathbf{\Theta}_k$ depends on its use in tree construction.

When a large number of trees is constructed, each of them produces a classification and they vote for the most popular class. These procedures are called random forests.

Breiman (2001) extensively describes the form of random forest that is formed by selecting at random (at each node) a small group of input variables to split on. The maximum tree is grown and no pruning is performed for each of the individual classifiers.

When performing bootstrap, roughly 37% of the examples in the learning set L do not appear in a particular bootstrap learning set $L^{(B)}$ (Breiman, 1996b). These can be employed to produce an *out-of-bag* estimation of the best number of random features participating in split induction without a necessity to perform cross-validation or allocate a dedicated test set.

The type of random forest described in Breiman (2001) is a process of bootstrap aggregation of decision trees that are built on randomly selected feature subsets, and the number of random features is obtained via an out-of-bag estimation.

Breiman (2001) shows that as more trees are added, random forests do not overfit but produce a limiting value of the generalization error. This upper bound of the generalization error can be derived in terms of two parameters that are measures of how accurate the individual classifiers are and of the dependence between them – strength and correlation. Split induction randomness maintains strength of produced classification rules (see also Section 6.1.6) and minimizes the correlation between various produced single decision trees. Bagging also seems to enhance accuracy when random features are used.

What are the major differences between bagging and random forests? Random forests do employ bagging, but the difference is not the sole introduction of random splits in tree induction. Recall that the original bagging procedure (Section 6.3.2) operates with *pruned* trees while random forests take maximum trees as inputs. This fact gives an

important speed advantage to random forests over bagged trees when pruning consumes significant amounts of time (for instance, when cross-validation is involved at any stage). Because of the *Law of Large Numbers*, when significant number of single classifiers are employed, random forests do not overfit, but a too small number of replicated samples during bootstrap would obviously not suffice. That means that if bagging had an option to have, say, only five replicated samples, doing so will violate the assumptions of the random forests methodology. Breiman (2001) employs 100 random forests in his empirical analysis to get *'reliable estimates'*.

Empirical comparisons in Breiman (2001) show that random forests perform much better than single trees. The average *absolute* misclassification rate improvement is around 12.5% for two various versions of random forests, which is obtained from 19 various data sets.

Forest-2	Forest-sel	Single tree
87.88%	87.31%	74.91%

Table 6.1: Average accuracy rates for: Forest-2 – a version of random forest with two random features, Forest-sel – a version of random forest with out-of-bag estimated number of random features, and a single tree classifier

The reported accuracy improvement is certainly quite significant, but being one of the ensemble methods, random forests suffer from the lack of interpretability. Although random forests offer a procedure that allows to compute variable importance via significance scores, which certainly may be helpful for some applications, it will not be possible to describe a produced decision rule in layman's terms as if it were a single axis-parallel classification tree.

6.3.4 Adaboost – Adaptive Boosting

Adaptive boosting or *Adaboost* (Freund and Schapire, 1997) is an ensemble machine learning technique that maintains a set of weights over the original learning sample and adjusts these weights recursively after a classifier is being analyzed – incorrectly classified observations gain higher weights.

Initially, equal weights are set for all observations in the learning sample. Let $t = 1, \ldots, T$ be the time index indicating the series of boosting rounds, $D_t(i)$ be the distribution of a set of weights on a training example i in round t. If there are n observations in the learning sample, for the first round $D_1(i) = \frac{1}{n}, i = 1, \ldots, n$. In each round, the weights of incorrectly classified examples are increased so that the single (or *weak*) classifier $h(x_i)$ is forced to focus on the hard examples in the learning sample (Freund and Schapire, 1999). A two-class problem will be considered here for simplicity so that $y \in \{-1, +1\}$, the generalization for multiclass problems is also possible.

The goodness of a weak hypothesis is characterized by its error ε_t that is measured with respect to the distribution D_t on which the weak learner was trained:

$$\varepsilon_j = \sum_{i=1}^{n} D_t(i) \mathbf{I} \left\{ h(x_i) \neq y_i \right\}. \tag{6.24}$$

In practice, either the weak learner can use the weights D_t on the training examples directly or the according set of training examples can be sampled in advance.

Once the weak classifier h_t was obtained, a parameter $\alpha_t \in \mathbb{R}$ is chosen. This parameter intuitively measures the importance that it assigns to the constructed classifier:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right). \tag{6.25}$$

Note that $\alpha_t \geq 0$ if $\varepsilon \leq \frac{1}{2}$ and α_t gets larger as ε_t gets smaller. The distribution D_t is then updated using the following rule:

$$D_{t+1}(i) = \frac{D_t(i)\exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t} \tag{6.26}$$

where Z_t is a normalization factor that is chosen so that D_{t+1} will be a distribution. The weight of examples misclassified by h_t is increased, the weight of correctly classified examples is decreased. The final classifier is constructed so that it is a weighted majority vote of the T weak hypotheses where α_t is the weight assigned to h_t:

$$H(x) = \operatorname{sgn}\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right). \tag{6.27}$$

where $\operatorname{sgn}(\cdot)$ is a sign function.

It can be shown that the training error of H is bounded (Freund and Schapire, 1997):

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}\{H(x_i) \neq y_i\} \leq \prod_{t=1}^{T} Z_t. \tag{6.28}$$

Instead of minimizing the training error of weak classifiers and consequently minimizing the training error of the final rule, Adaboost finds Z_t in each round that can be done by selecting optimal weights α_t and h_t. While weak classifiers h_t are exogenous for the system, α_t can be optimized:

$$Z_t = \sum_i D_t(i)\exp\{-\alpha_t y_i h_t(x_i)\} \to \min$$

$$\frac{dZ}{d\alpha} = 0 \Rightarrow -e^{-\alpha}(1-\varepsilon) + e^{\alpha}\varepsilon = 0 \Rightarrow \alpha = \frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right),$$

which corresponds to the equation (6.25).

Having $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$,

$$Z_t = \sum_i D_t(i)\exp\{-\alpha_t y_i h_t(x_i)\} = (1-\varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)},$$

and Z_t is minimized by selecting h_t with the minimum weighted error ε_t, which justifies the weighted error minimization.

An *implied* assumption of Adaboost is that weak classifiers have the classification

errors that do not exceed 50%, i.e. weak classifiers are better than random:

$$\epsilon_t \leq \frac{1}{2}. \tag{6.29}$$

If the condition (6.29) is violated, the adaptive procedure effectively halts, and many modern versions of Adaboost have explicit condition (6.29) included. When weak classifiers with $\epsilon_t > \frac{1}{2}$ were tried to be used with Adaboost in Breiman (2001), the procedure could not be finished successfully.

Quinlan (1996) shows that boosting implicitly requires instability, otherwise, given the same subsequent classifiers, the weight adjustment scheme has the property that $\epsilon_t = \frac{1}{2}$, and all following classifiers have votes with the zero weight in the final classification.

Forest-2	Forest-sel	Adaboost	Single tree
87.88%	87.31%	87.20%	74.91%

Table 6.2: Average accuracy rates for: Forest-2 – a version of random forest with two random features, Forest-sel – a version of random forest with out-of-bag estimated number of random features, Adaboost – boosted version of single tree classifiers, and a single tree classifier

There are certain links between Adaboost and other machine learning methods. Adaboost can be regarded as a special case of random forest if the weights on the training set are selected properly, see Breiman (2001) for more details. There is a certain relation to SVMs, however, notable differences do exist, too, refer to Freund and Schapire (1999) for a broader discussion.

Empirically, Adaboost shows comparable results with different versions of random forests in terms of the error rate (Breiman, 2001). Table 6.2 expands the numeric evidence from Section 6.3.3 by adding an adaptively boosted version of the single tree classifier.

While there seems to be no significant difference in the accuracy between Adaboost and random forests, Breiman (2001) reports that random forests work much faster: *'growing the 100 trees in random forests was considerably quicker than the 50 trees for Adaboost'.* For industrial applications that may be not too important, but speed factor is certainly crucial for other settings (e.g. academic).

6.3.5 Ensemble Methods and Stock Picking

Ensemble methods are certainly a step forward in obtaining classification models with the reduced misclassification rate compared to the setup when single classifiers are employed. Theoretical properties of random forests and Adaboost provide rigorous conditions when these ensemble methods are capable of producing superior results in terms of accuracy.

However, as it was pointed out in Section 6.2.7, accuracy is certainly an important but not the only performance measure of various classification rules (refer also to an example in Table 8.4). Random forests and adaptively boosted trees are black box rules, they can not be explained in layman's terms. It is true that random forests can provide variable significance information (Adaboost and bagging lack this useful

feature), however the attained level of interpretability will just be comparable to that of, say, factor models. Single axis-parallel decision trees clearly supersede this level. Section 3.1 explicitly outlines model requirements, and on this basis ensemble methods can not be considered further *in the particular setup of this work*.

Many hedge funds, for instance, employ various aggressive algorithmic trading strategies, especially with high-frequency data. Investment decisions should be carried out automatically there with little or no chance to analyze individual market situations. Ensemble methods are excellent candidates for this role.

The way how investment decisions are carried out brings the next important property – execution speed. For instance, Adaboost is a comparatively slow procedure, and even for some industrial applications where input information and classifications are updated online, that may become a limiting factor. The execution speed is one of the crucial factors in this study, too, because the computing power available at the moment is rather limited.

For large data sets, Dietterich (2000a) concludes that node variables randomization can generally be expected to perform better than bagging since bootstrap replicates of a large sample are very similar to the learning sample itself, and the obtained decision rules are not very diverse (their correlation is too high in terms of the random forests methodology). In high-noise setups, it is concluded that Adaboost puts a lot of weight on mislabeled observations that leads to serious overfitting problems. Random forests, on the contrary, do not overfit, and according to Breiman (2001), this ensemble technique is relatively robust to outliers and noise: *'Adaboost deteriorates markedly with 5% noise, while the random forest procedures generally show small changes'* because random forests do not place weights on different observations. At the same time, Breiman (2001) acknowledges that noise issues of Adaboost are data set dependent. Dietterich (2000b) compares randomization, bagging, and boosting and concludes that when there is no or little noise, boosting shows the best results and randomization is competitive with bagging. However, when the data noise is more severe, bagging is clearly superior to boosting and sometimes better than randomization, which generally supports the empirical findings from other studies.

The issue of stability is one the key factors that stays beyond a novel pruning technique – one of the key points of this study – that is to be introduced next.

Overall, given the empirical comparisons of various popular ensemble methods, one may conclude that random forests – a combination of the feature randomization and bagging – may be an excellent choice for black box trading algorithms where no model parameters (like the rule size) are to be selected. For applications with higher desired levels of interpretability, random forests and other ensemble methods are not a feasible choice, but there are some other advantageous properties of single classifiers that are discussed below.

7 Best Node Selection – Novel Way of Tree Pruning

7.1 Pruning Structures – Node Triplets and Individual Nodes

By its architecture, the majority of the aforementioned pruning algorithms including the cost-complexity approach ultimately operates with triplets of nodes $\{t_P, t_L, t_R\}$, which are parts of optimized subtrees of T_{MAX}. The decision whether to employ the selected triplet or not is based on the *joint performance of two child nodes in the triplet*, refer to Section 5.3 (definition of the 'weak link') and Section 6.2 for more details.

However, there are many cases when only one of the child nodes contains homogenous data while the second one is filled with points belonging to various classes. Performing validation (or computing one of the quality indices) of the subtree containing both child nodes, which is done traditionally, frequently results in the mediocre performance of the triplet as a whole.

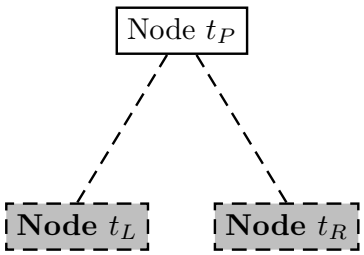

Figure 7.1: Traditional CART pruning operates only with both child nodes simultaneously – both child nodes are pruned here

Hence there are serious reasons to consider that at least for selected types of classification tasks containing inhomogeneous data (which is true, for instance, for stock picking), the traditional cost-complexity and other similar pruning approaches may not provide the best feasible results for single tree classifiers.

Best Node Selection (BNS) analyzes individual node performance and provides an opportunity to prune only one child node if necessary, at the same time pruning both child nodes simultaneously (as it is done traditionally) is also an option.

While cost-complexity pruning relies solely on the cross-validation performance of a given subtree, the 'quality' of individual nodes is ignored. The tree 'quality' is estimated via an *integral* characteristic as defined in the equation (5.5), therefore individual nodes have only a minor impact on the overall result.

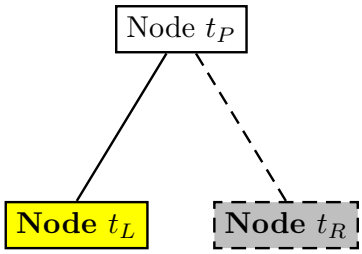

Figure 7.2: Situation that is infeasible for the traditional cost-complexity approach – only one child node is pruned here

7.2 Two Measures of Node Reliability

BNS reverses this approach and assumes that performance of the tree is driven by *individual* performance of tree nodes. This novel method of tree pruning takes into account *only* individual node characteristics and does not perform cross-validation to obtain an integral measure of the tree performance.

For a classification tree with J classes in the learning sample, define the *dominating class* j^* in a node t as the one containing the biggest number of observations: $j^* = \text{argmax}_i\, p(i|\, t)$. BNS employs two quality measures that assess any individual tree node:

- *node purity* $\bar{p}$: node purity measures the proportion of a dominating class in a node,
- *node size* $\bar{n}$ *in terms of a dominating class*: the second node quality measure assesses a given node in terms of representativity.

This can be best illustrated using the following examples as well as justification of nonsymmetrical pruning as shown on Figure 7.2.

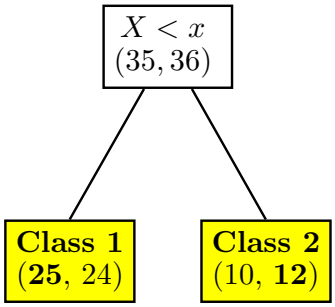

Figure 7.3: Illustration of the *node purity* criterion of BNS. Numbers in parentheses indicate the number of cases for the first and second classes

Consider a subtree containing two terminal nodes on Figure 7.3. Recall that class tags for terminal nodes are assigned according to the majority principle (refer to the equation (4.8)), and in this situation class tags have been assigned unambiguously. Note that in situations when the maximum in (4.8) is not unique, class tags are assigned

randomly (although randomness is limited to the set of dominating classes). However, does the situation on Figure 7.3 differ so much from an outcome when class tags have to be assigned randomly? To answer this question, the internal misclassification errors (see Section 5.3) should be computed for both nodes. For the left terminal node, the internal misclassification error is $\frac{24}{24+25} = 48.98\%$. For the second terminal node, the error is $\frac{10}{10+12} = 45.45\%$. The internal misclassification error shows the probability of making the wrong classification using a particular node when the structure of the learning sample is assumed to be unchanged. The calculated probabilities are really high in this example because the values are very close to 50%, a situation when the classification choice is made completely randomly.

BNS provides the end-user with controls to define the comfortable threshold value of the internal misclassification error enabling the careful selection of parts of a decision rule that lead to desired risk characteristics. This can be achieved by setting a user-defined threshold value $\bar{p}$. As it could be seen from Section 6.2.7, the node purity criterion reminds a part of the discriminatory power measure from Osei-Bryson (2007) with a key difference that the measure is computed for the whole tree as the weighted sum of posterior probabilities, and BNS aims at characterizing each node individually.

The second crucial element of BNS is to ensure the use of only *representative* parts of a classification rule by controlling the number of observations belonging to a dominating class in each evaluated node. The motivation for the auxiliary criterion is the following. Consider the example on Figure 7.4 where the right child node contains only three observations, and the proposed classification is based only on two observations of the second class or 2.8% of the size of the learning sample. If the right child node is analyzed only from the position of purity, then the misclassification risk of 33% may be considered acceptable for many applications. However, the sole measure of risk does not show the whole picture, and one can clearly see that the right child node does not contain a large chunk of data and therefore is very unlikely to uncover some fundamental link of the underlying data. Instead, one may conclude that it is likely that the right terminal node only filters outliers or noise, both of which are not consistent over time.

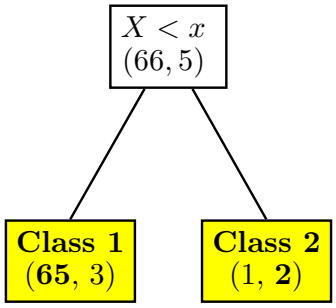

Figure 7.4: Illustration of the *node representativity* criterion of BNS. Numbers in parentheses indicate the number of cases for the first and second classes.

The key idea of BNS is, therefore, to analyze the *reliability of nodes* in terms of their *purity* and *size* and allow nonsymmetrical pruning when necessary. If a terminal node, which is potentially a crucial element of the final classification of new data, contains a mixture of observations belonging to various classes, its presence in the decision rule is

desirable only if one class clearly dominates others because otherwise the reliability of the classification decision may be compromised.

At the same time, BNS employs the idea that each node after pruning should not contain only a minor number of observations from the learning sample. Put differently, each part of the classification rule should be representative. The limitation of the node size as one of the ways of effective tree pruning is also mentioned in Schroders (2006).

7.3 Reject Option and BNS

If the data to classify fall into any of unreliable nodes, i.e. nodes with values of $\bar{p}$ and $\bar{n}$ that are out of the user-defined range (see below), BNS effectively switches to a reject region of a given classification rule. *Reject option* is a commonly adopted method to reduce the classification error when the underlying structure is assumed to change over time or input data are noisy, and errors are, therefore, unavoidable. The basic principle behind reject option is that when it is not confident for a classifier to label a certain observation, such an observation should not be given a class tag, and potential misclassifications are converted into rejection. Chow (1960) points out that the tradeoff between the errors and rejections is seldom one to one, and it is possible that some potentially correct recognitions are also converted into rejections. However, many applications employing classification with reject options appear to be more successful than their traditional counterparts, see for example Ahmadi et al. (2004) or Hanczar and Dougherty (2008).

Reject option intentionally allows a decision tree to classify not all of the examples. As it is emphasized in Hanczar and Dougherty (2008), when all examples are classified, the classification rule has no control over the accuracy of the classifier because a traditional technique just aims to produce a classifier with the smallest error rate possible. This setup corresponds to cost-complexity and other popular aforementioned methods of tree pruning. BNS introduces reject option based on two reliability measures – node size and purity – and provides the end-user not only with an ability to influence the target internal misclassification error, but also with the means to directly control the representativity of the rule.

Chow (1960) shows that there is a general relation between the misclassification error and rejection rate: the error rate decreases monotonically while the rejection rate increases. Based on this result, Chow (1960) proposes an algorithm that finds an optimal classifier that balances optimal error and reject tradeoff. However, an optimal classifier can be found only if the true posterior probabilities are known, which is seldom the case in practice. Hanczar and Dougherty (2008) provide a short overview of alternative methods, however all of them are ultimately assessed by the accuracy rate. While that may be reasonable for some applications, applied economic tasks like stock picking are more likely to be evaluated in terms of yields or generated profit, which is a completely different approach where each classification has different value that is proportional, say, to the relevant market volatility. Classification rule assessment based solely on the accuracy rate may be only of academic interest in such a setup since a rule with lower misclassification rate may return lower yield due to lower price magnitude and maximum absolute values for correctly classified data, see Section 8.2 and Table 8.4 for more details.

7.4 Rigorous Formulation of BNS and Its Properties

The more balanced control – individual node vs. node triplet – comes at the cost of introducing two degrees of freedom. Let $\bar{n}$ be the minimum required number of observations of the *dominating class* j^* in a node t, $j^* = \text{argmax}_i\, p(i|\, t)$, and $\bar{p}$ – the minimum required proportion of the dominating class j^* observations. Assuming that there are J classes in the learning sample, if the following conditions hold:

$$\begin{cases} n_{j^*}(t) \geq \bar{n}, \\ \frac{n_{j^*}(t)}{n(t)} \geq \bar{p} \geq \frac{1}{J}, \end{cases} \tag{7.1}$$

then the node t is called *reliable* and marked as such via the *boolean function* $v(t) = 1$, which takes the zero value if (7.1) does not hold. If n_{LS} is the size of the learning sample, the feasibility constraint for (7.1) is straightforward:

$$\begin{cases} \frac{\bar{n}}{\bar{p}} \leq n_{LS}, \\ \bar{p} \leq 1. \end{cases} \tag{7.2}$$

Let $T(n)$ be the tree where each terminal node contains *at most* n observations unless all observations at the node belong to the same class. To obtain a classification decision for a given new observation $x \in \mathbb{R}^p$ using BNS, it suffices to build the tree $T(\frac{\bar{n}}{\bar{p}})$, locate the terminal node $t[x]$ of the observation x, and check if this node is reliable. If it is, then the optimal decision is the one produced by that node. If not, then assuming that the Gini index is employed as the impurity function, all parent nodes of the given node $t[x]$ are also unreliable, therefore tree pruning results in an empty tree, and *it is advised by the BNS procedure to perform no classification based on the provided tree as chances for misclassification are considered to be rather high.* This can be called the *inverse propagation property* of BNS: if a child node is unreliable, its parent node is unreliable as well. Put differently, the reject region of $T(\frac{\bar{n}}{\bar{p}})$ ensures that all parent nodes of children from the reject region will be rejected, too, if child nodes are pruned.

These two statements about the optimal tree size $\frac{\bar{n}}{\bar{p}}$ and inverse propagation property need to be proven, of course. First of all, let us consider an arbitrary maximum tree $T_{MAX} = T(1)$ with exactly one observation at each terminal node. For a given observation $x \in \mathbb{R}^p$ to classify, the terminal node $t[x]$ of $T(1)$ is unreliable unless $\bar{n} = 1$ or, if $\bar{n} > 1$, the condition $n(t[x]) > \bar{n}$ is violated since $t[x] \in \tilde{T}(1)$ and $n(t[x]) = 1$ where $\tilde{T}(1)$ is the set of terminal nodes of a tree $T(1)$. It may happen though that $p(j^*|t[x]) = 1$ and $n(t[x]) > 1$. Then still if $n(t[x]) < \frac{\bar{n}}{\bar{p}}$, the node is unreliable because either the condition $n(t[x]) \geq \bar{n}$ is violated from (7.1) or, if not, with the minimum required probability of the dominating class being equal to $\bar{p}$ and number of observations of that class being equal to $\bar{n}$, the *minimum* feasible terminal node size not to violate (7.1) is $\frac{\bar{n}}{\bar{p}}$.

Therefore, if BNS-reliable nodes exist in T_{MAX}, *all of them* can be reached by building the tree $T(\frac{\bar{n}}{\bar{p}})$. The proof of Theorem 1 (in the Appendix) is in fact the proof of the inverse propagation property of BNS. Hence if $t[x] \in \tilde{T}(\frac{\bar{n}}{\bar{p}})$ is unreliable, then all the parent nodes of $t[x]$ are unreliable, too.

At the moment, the inverse propagation for an *arbitrary* impurity function (and not just for the special case of the Gini index) is proven only for the case when dominating

classes of the child and parent nodes coincide, refer to Lemma 2 (in the Appendix) for details. However, as it was mentioned above, since the choice of the impurity function does not change the configuration of the maximum tree a lot (refer to Section 6.1.6), this creates little or no limitations in practical applicability of the method depending whether there is a rigid constraint to employ a particular form of an impurity function. Even if there is one and an impurity function different from the Gini index must be used, that would only mean that all terminal nodes and their parents, if necessary, should be checked for the condition (7.1) assuming that (7.2) holds.

To conclude, the traditional cost-complexity approach builds a maximum tree at the first step. Then a sequence of subtrees is found by the minimization of the cost-complexity function. The last step is to find the optimal tree by employing cross-validation. There, a set of cost-complexity estimates for different subtrees is found and a rule of thumb is applied to select the optimal tree. Other aforementioned tree pruning methods compare either various node triplets of T_{MAX} exhaustively or select subtrees of T_{MAX} based on a certain criterion. The optimal tree is found as the one optimizing an integral measure of quality. On the other hand, BNS requires to build the tree $T(\frac{\bar{n}}{\bar{p}})$ using the Gini index. After that an observation to classify is to be processed by the tree in the following way: the decision rule produced by $T(\frac{\bar{n}}{\bar{p}})$ is valid only if the respective terminal node is reliable as indicated in (7.1) and (7.2). Otherwise, the rejection zone is reached and it is suggested to avoid conducting classification using the available tree as chances for misclassification are considered to be rather high. In the stock picking setup that would be equivalent to taking a neutral position.

7.5 Applications of BNS to Noisy Data Sets

Applications of BNS are not solely limited to severe cases like those on Figure 7.3 and Figure 7.4 and likely to show their bests properties on data sets with the high level of noise.

Let us consider a slightly modified example from Figure 4.1 and introduce some overlapping of the elements in a three-class problem depicted on Figure 7.5. Classes *black* and *green* are not linearly separable anymore, and that creates a challenge for the canonical cost-complexity approach, which is not able to keep only one of the child nodes in the decision tree.

Now, let us consider the tree produced by the cost-complexity approach (the left one on Figure 7.6) and its terminal node with the class tag *black*. It partially corresponds to the mixed area on Figure 7.5 where observations of both *black* and *green* classes are present. It contains 68 points of the class *black* and only 12 points of the class *green*, and the risk of making the wrong classification decision, all other things being equal, is $\frac{12}{12+68} = 15\%$. The aim of BNS is to reduce or, when possible, to avoid fully this risk – BNS results in a slightly different tree (the right one on Figure 7.6) where another perfectly homogenous cluster of points is separated, which corresponds to the auxiliary condition $X_1 < 0.75$. Setting up $\bar{p} = 75\%$, when the data to classify appear to be in the unreliable node of the BNS tree (put differently, when these data meet the conditions $X_1 \geq 0.75$ and $X_2 \leq 0.5$), the decision rule considers this area of the learning sample unreliable (the risk of misclassification, all other things being equal, is already $\frac{12}{40} = 30\%$) and suggests no reliable classification can be performed. In the realm of stock

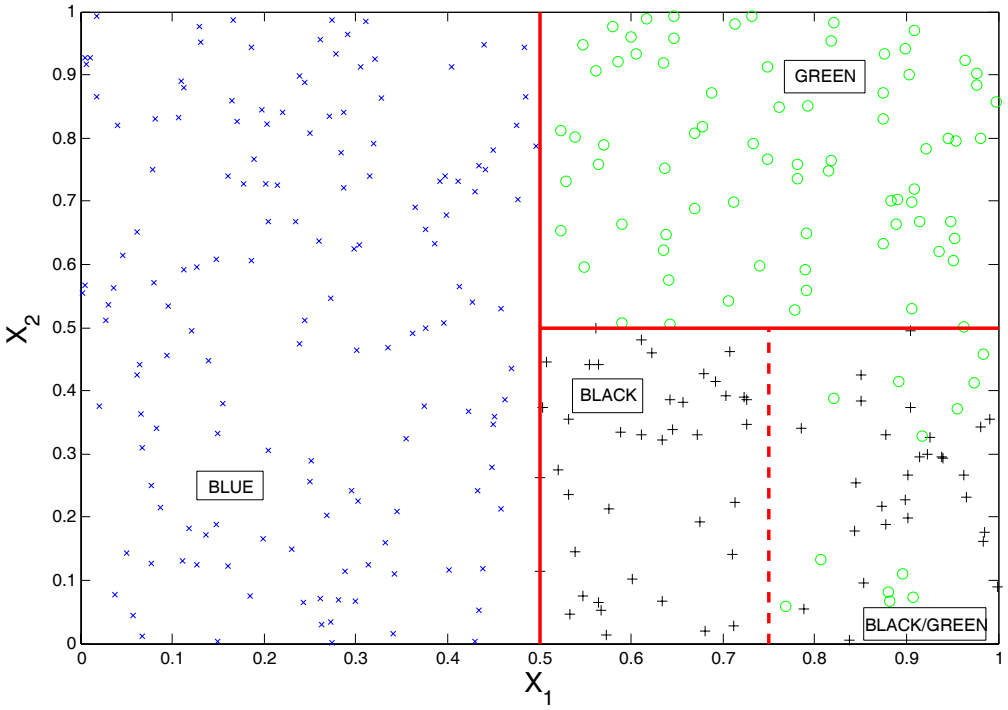

Figure 7.5: Solid lines refer to recursive partitioning suggested by the canonical cost-complexity approach, the dashed line indicates another partitioning that is missing and might be useful to separate a lot of points belonging to the class *black*.

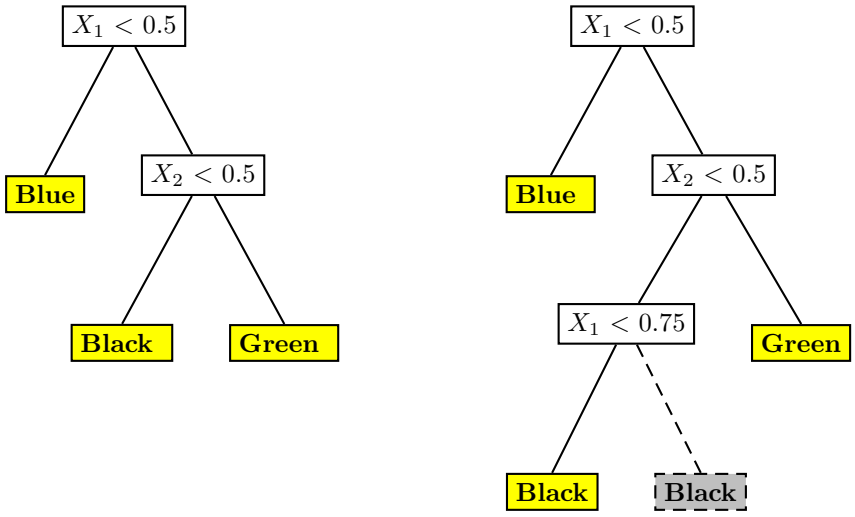

Figure 7.6: Two trees produced by the cost-complexity approach (left) and novel Best Node Selection (right). The grey dashed node of the tree to the right indicates the noisy part of the data in the learning sample and suggests this cluster of the decision rule to be excluded when classifying new data (reject region).

picking that would mean the recommendation to take the neutral position. On the other hand, the tree produced via the cost-complexity approach is not able to differentiate between two terminal nodes with the same class *black* (as on the tree produced by BNS) that implies taking the extra risk of misclassification. This happens because only symmetrical pruning is available for the cost-complexity approach.

Certainly, different settings for reject areas can be employed, and there are at leat two options available. Optimal parameter values can be derived via calibration (see Section 8.4). Another way is when the end-user, following the prespecified risk preferences, sets up the appropriate values directly. However, similar to the empirical 1-SE rule of thumb of cost-complexity pruning proposed in Breiman et al. (1987) (refer to the equation (5.16) and Section 5.3 for more details), the following rule of thumb can be used for the selection of BNS parameters: the node can be called reliable if observations of the dominating class in the node account for at least 75% of the node size and at least 10% of the learning sample size. The threshold probability value $\bar{p}$ of 75% corresponds to the natural choice of the relevant value of the discriminatory power measure in Osei-Bryson (2007), and the size parameter $\bar{n}$ of 10% of the learning sample size seems to a reasonable value for many applications. The tree on Figure 7.6 employs this empirical rule of thumb.

In some cases a highly-noised learning sample contains small islands of relatively homogenous data that are quite difficult to extract in terms of axis-parallel splitting, which results in quite sophisticated decision rules. In the presence of other noisy data chunks in surrounding nodes, traditional pruning methods are very unlikely to keep these distant nodes because they compromise the validation score of the near subtrees that include mostly inhomogeneous data islands.

The question is certainly whether in practice similar setups are likely to appear where complicated decision rules have the advantage of describing selected data islands while keeping the rest parts of the rule in the rejection zone. Figure 7.7 is an excellent illustration of this phenomenon that backs the advantageous applicability of BNS (at this point – at least at the theoretical level) to real financial data. This example also employs the rule of thumb for selecting BNS parameters and clearly demonstrates the ability of novel pruning to reach distant data islands in terms of complexity of the rule. Classical pruning techniques are very likely to contain only one of three reliable nodes (at the top of the tree) and force the final decision rule to have mediocre risk characteristics for the left part of the tree. Empirical comparison of BNS and traditional cost-complexity pruning on DAX30 data set is performed in Chapter 8.

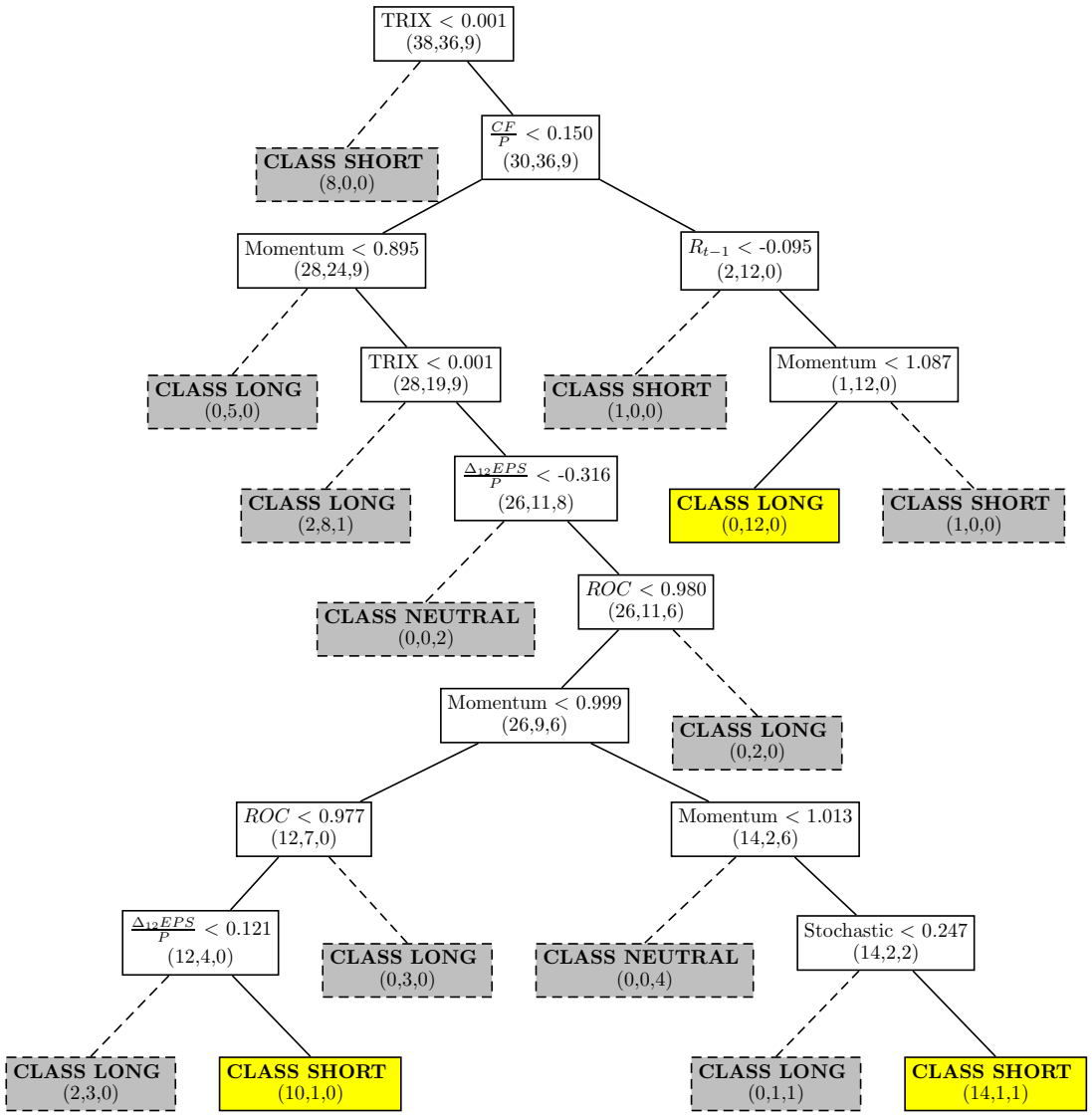

Figure 7.7: Practical illustration of how BNS works when applied to the learning sample with lots of noise (BAS stock) – only three terminal nodes contain reliable parts of the decision rule

8 Historical Simulation of DAX30 Stock Picking

8.1 General Setup and Available Data

Having analyzed a lot of various theoretical and practical issues of the stock picking challenge and particularly classification tree induction and pruning techniques, the next important step is to implement practically a trading strategy that takes available market data as the learning sample and produces trading recommendations. Looping this procedure over time, it is of particular interest to compare historically simulated risk-return characteristics of trading algorithms based on cost-complexity pruning and BNS. The choice of cost-complexity pruning is consistent with the analysis in Section 6.2.8 and choice of the default pruning method in Murthy et al. (1994).

The available stock market data for the analysis (DAX30 German stock index) consist of the samples for 15 companies and for the time period of February 19, 2001 – May 31, 2004. 13 different fundamental and technical variables were at the disposal describing each of these companies, refer to Table 8.1 and Table 8.2 for more details. The time scale of the data is one week.

Indicator	Company Name
ADS	ADIDAS-SALOMON AG
ALT	ALTANA IND-AKTIEN U. ANL AG
ALV	ALLIANZ AG
BAS	BASF AG
BAY	BAYER AG
BMW	BAYERISCHE MOTOREN WERKE AG
DCX	DAIMLERCHRYSLER AG
EOA	E.ON AG
LHA	DEUTSCHE LUFTHANSA AG
LIN	LINDE AG
MAN	MAN AG
SAP	SAP AG
SCH	SCHERING AG
SIE	SIEMENS AG
TUI	TUI AG

Table 8.1: List of companies from the DAX30 index and their codes

Because the DAX30 index consists of stocks of companies with different business backgrounds, one of the novel features of this of work is to analyze each company individually as opposed to pooling learning sample data, which is done traditionally, see

Sorensen et al. (1999) or Seshadri (2003) for more details. Put differently, each stock recommendation is to be derived via an individual classification tree, which is optimized independently. This is done to allow for potential heterogeneity of different DAX30 components.

Historical simulations were run, in fact, twice: when cost-complexity pruning is employed to derive optimal trees and when pruning was switched to BNS. All crucial calibration elements like the class assignment rule and others coincide for both competing trading strategies so that an adequate empirical comparison could be carried out later.

The available market data were employed in the following way. The first 53 observations (or roughly one year) were allocated to the learning period. The next 25 points (or roughly half a year) comprised the test set for calibration (Section 8.4). Finally, the rest 93 points (or a little less than two years) were used for backtesting (Section 8.5). The size of the sliding window, when applicable, was set to the length of the learning period – 53 observations.

8.2 Class Assignment Rule and 'Big Hit' Ability

Decision trees are built from the learning samples $(\boldsymbol{X}, Y)$ where $\boldsymbol{X} \in \mathbb{R}^p$ is the matrix of explanatory variables for *each* stock. Because the ultimate goal of the described procedure is to introduce an effective strategy of stock picking, a vector Y should represent a characteristic of stock profitability over time. In this work, one-period returns are of special interest since it is supposed that each new period (week) the trading strategies are updated with the new incoming market data and realized current stock prices. Having stated that and recalling the equation (3.1), the proposed choice of Y here is a *forward-looking* one-period stock yield.

Therefore, for each stock independently, the natural choice of the class assignment rule could be the value of

$$\operatorname{sgn}(R_t) = \operatorname{sgn}\left(\frac{P_{t+1} - P_t}{P_t}\right) \tag{8.1}$$

where $\operatorname{sgn}(\cdot)$ is a sign function and P_t is the current stock price. However, it is possible that other nonzero (positive) threshold values may lead to better forecasting results when only relatively substantial stock price changes are indicated as those pointing to presently under- or overvalued stocks. The *'big hit'* ability or the ability of a method to forecast effectively the movements with big relative magnitude is introduced and thoroughly discussed in Hartzmark (1991).

The first model degree of freedom is, therefore, a threshold value $\bar{R} \geq 0$ that defines the class y of each stock for a given next period return in the learning sample. Note that for new data to classify, next period stock returns are not available and, in fact, are objects of forecasting. Depending on the next period stock performance, there are three classes employed: *long*, *short*, and *neutral*. $\bar{R}$ can potentially have different values

for different calibrated stocks allowing for necessary stock differentiation:

$$\begin{cases} R_t > \bar{R},\ y_t = \{long\}, \\ R_t < -\bar{R},\ y_t = \{short\}, \\ -\bar{R} \leq R_t \leq \bar{R},\ y_t = \{neutral\}. \end{cases} \tag{8.2}$$

Note that from the equation (8.2) it directly follows that setting $\bar{R} = 0$ is equivalent to the intuitive setup when even the smallest (and possibly insignificant or improperly measured) change in the stock price triggers the assignment of active positions (long or short) to respective stocks. Because $\bar{R} \geq 0$, the proposed algorithm does not restrict a procedure from setting $\bar{R} = 0$ for some of the stocks and provides more flexible calibration opportunities.

Given the magnitude of weekly stock returns in the learning sample, $\bar{R}$ was selected for each stock independently during calibration from the grid $[0\%, 3\%]$ with the step of 0.5%, refer to Section 8.4 for more details.

8.3 Input Variables and Types of Learning Samples

Although a decision tree chooses variables for the learning sample automatically when building a classification rule, there is always a possibility for spurious links between the dependent and independent variables or instability of data structures over time. In Section 6.3 it was pointed out that classification trees are more likely to be unstable over time than some other methods like the k-nearest neighbor classifier, and on this basis two different input specifications for the learning sample – basic and extended – are considered.

Unlike Brennan et al. (1999), where the preliminary regression analysis of available data was supposed to find the most significant variables to be included in the tree(s), in this study the optimal specification for each stock was obtained from a calibration procedure, which is described below.

Two configurations of $\boldsymbol{X}$ were considered for each stock independently. The first one mostly resembles the ideas of fundamental analysis (Fama and French, 1992; Sorensen et al., 1999; Brennan et al., 1999) and therefore is based on variables of the fundamental nature – these are listed in the upper part of Table 8.2. According to the first specification, the learning sample consists of four variables: $\frac{CF_t}{P_t}$, $\frac{EPS_t}{P_t}$, $\frac{\Delta_{12}EPS_t}{P_t}$, and ROE_t (t is the current time period and $\frac{\Delta_{12}EPS_t}{P_t}$ is a three-month/12-week relative change of EPS).

If the basic configuration of the learning sample passes successfully the calibration procedure (described in Section 8.4), the Occam's Razor principle is applied and no auxiliary input variables are added. However, if that is not the case, the basic specification is augmented by additional features listed in the lower part of Table 8.2. These variables are backed by the technical analysis methodology of stock price forecasting, see Neftci (1991) and Sullivan et al. (1999) for a broader discussion of technical analysis and its relevant statistical applications. The second degree of freedom is, therefore, the type of specification.

Depending on the stability of a distribution and the level of noise of the learning sample over time, retaining the old observations in the learning sample may potentially result in a deteriorated forecasting power of the model (Tam and Kiang, 1992); there-

Indicator	Type	Frequency	Comments
Sales/P	Fundamental	1 week	Sales to Price Ratio
CF/P	Fundamental	1 week	Cash Flow to Price Ratio
EPS/P	Fundamental	1 week	Earnings per Share to Price Ratio
Δ_{12}EPS/P	Fundamental	1 week	3-Month Change in EPS to Price Ratio
ROE	Fundamental	1 week	Return on Equity
Momentum	Technical	1 week	$M_t = P_t - P_{t-T},\ T = 20$
Stochastic	Technical	1 week	$\frac{P_t - P_L}{P_H - P_L},\ P_H = \max(P_t),\ P_L = \min(P_t)$
MA/P	Technical	1 week	$MA(T) = \frac{\sum_{i=t-T}^{t} P_i}{T},\ T = 12$
MACD	Technical	1 week	$(1 - \frac{n_1}{n_2})\{MA(n_1) - MA(n_2 - n_1)\}$ $n_1 = 12,\ n_2 = 26$
MA St. Error	Technical	1 week	Standard deviation of MA
ROC	Technical	1 week	$ROC_t = \frac{P_t}{P_{t-T}},\ T = 10$
TRIX	Technical	1 week	Triple exponentially smoothed MA
R_{t-1}	Technical	1 week	$R_{t-1} = \frac{P_t - P_{t-1}}{P_{t-1}}$, P_t – current stock price

Table 8.2: List of available variables as potential input factors for learning samples. All variables are available for each of the 15 analyzed companies. The current time period is indicated by t

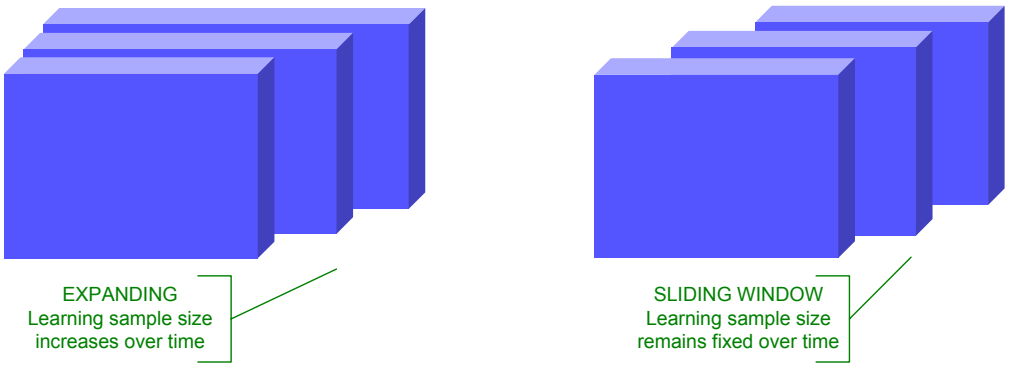

Figure 8.1: Two types of the learning samples employed

fore, the third degree of freedom for calibration is the type of the learning sample, which can either have the fixed size over time (*sliding window*) or, when such a setup provides inadequate calibration results (see below), include each new available observation with the following step (*expanding learning sample*). As in the case of input variable specifications, sliding window (being a simpler structure) is tested first and kept if adequate calibration results are achieved.

8.4 Parameter Calibration

For each stock independently, the adequacy of calibration was assessed primarily based on the expected annualized yield – the higher is the yield, the better is assumed to be the specification. As it has been mentioned above, the accuracy measure is highly suitable

Stock	Specification	$\bar{R}$	Learning sample
ADS	fund. and tech.	0.5%	sliding window
ALT	fundamental	1.0%	sliding window
ALV	fundamental	1.0%	sliding window
BAS	fund. and tech.	0.5%	expanding
BAY	fundamental	0.5%	sliding window
BMW	fundamental	1.0%	sliding window
DCX	fundamental	1.0%	sliding window
EOA	N/A	N/A	N/A
LHA	N/A	N/A	N/A
LIN	fundamental	0.5%	sliding window
MAN	N/A	N/A	N/A
SAP	N/A	N/A	N/A
SCH	fundamental	0.5%	expanding
SIE	N/A	N/A	N/A
TUI	fund. and tech.	0.5%	expanding

Table 8.3: Calibration results for BNS tree pruning, N/A indicates situations when none of the inputs were able to produce a positive calibration yield

for many academic applications. However, in trading environments the financial result is what ultimately counts. While chances of a more accurate procedure to produce better financial results are higher, all other things being equal, a superiority of one procedure over another in terms of only the accuracy measure does not guarantee its superiority in terms of financial returns. A simple example can support this statement: suppose that a procedure A shows the 80% accuracy rate capturing 1% stock price changes correctly every time and making two wrong 5% stock price change classifications, and a procedure B yields only 60% of correct classifications (see Table 8.4). However, showing the 'big hit' ability, the procedure B classifies correctly all 5% price changes. Which procedure shows better results? For stock picking applications, all other things being equal, the procedure B supersedes the procedure A since the overall absolute profit is 26% (procedure B) vs. 20% (procedure A) or more than a 30% relative difference. In the ideal situation, high accuracy rates would combine with the high level of yielded profits.

Stock Price Change	1%	1%	1%	1%	1%	5%	5%	5%	5%	5%
Procedure A	•	•	•	•	•	•	•	•	○	○
Procedure B	○	○	○	○	•	•	•	•	•	•

Table 8.4: Two classification procedures making either hits (•) or misses (○) when forecasting sequential stock price changes. Procedure B exhibits a lower hit rate but superior financial result

To avoid potential spuriousness of calibration results, several countermeasures were employed. Define an active classification operation as the one yielding classes long or short. Firs of all, the *activity ratio* indicator (the percentage of active operations during calibration for a given stock) was used in the following way. The activity ratio had to exceed 40% in order for a specification to be considered reliable, and competing

specifications (with the similar amount of the yielded expected return) were selected in favor of those with the highest activity ratio. 40% is a subjective reasonable value, and it simply cancels those specifications that are unable to provide at least 40% of active classifications during the calibration period (see Section 8.5), which may happen when either $\bar{R}$ is set too high or, say, BNS parameters $\bar{n}$ and $\bar{p}$ are set inappropriately.

Additionally, the *hit ratio* (the proportion of correct active directional forecasts during the calibration) of a reliable specification had to exceed 45%, which roughly corresponds to a similar condition of the weak classifier construction for various ensemble methods, see Section 6.3 for more details.

Finally, the *representativity ratio* (the average proportion of observations belonging to two active classes in the root nodes of trees during the calibration period) had to exceed 50% to ensure that the joint proportion of classes long and short is not too low. This constraint is applied primarily to avoid spuriousness of calibration results due to unreasonably high values of $\bar{R}$.

Therefore, a specification can pass the calibration phase if and only if:

- accumulated profit by the end of the calibration period is positive;
- activity ratio exceeds 40%;
- hit ratio exceeds 45%;
- representativity ratio exceeds 50%.

If two specifications show close results in terms of the accumulated calibration profit, a specification with the higher activity ratio is preferred. If one specification demonstrates a significantly higher level of profit (at least 1.5 times higher) and passes all four necessary calibration conditions, it is preferred to the other specification.

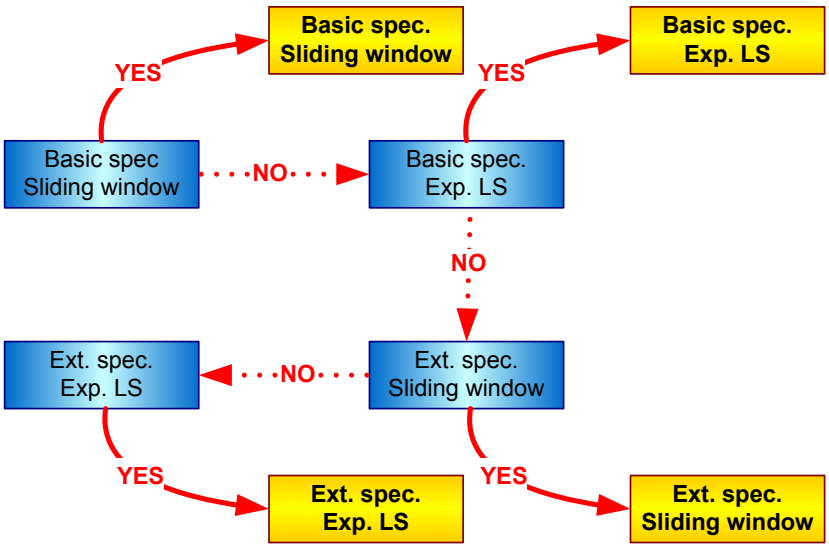

Figure 8.2: Specification priorities according to the Occam's Razor principle and calibration process

Figure 8.2 illustrates the priority of various specifications that are comprised of either basic or extended feature sets and have the learning sample with either fixed (sliding window) or expanding size. For each stock individually and for fixed $\bar{p}$ and $\bar{n}$, calibration starts with selecting the appropriate value of $\bar{R}$ for the sliding window learning sample constituted by the basic feature set. If such $\bar{R}$ can not be found, the more complex learning sample type is employed, etc. If at the end of calibration no appropriate $\bar{R}$ can be obtained even for the case of the extended feature set and expanding learning sample, the analyzed stock is excluded from the portfolio (see Section 8.5).

In the case when all four configurations from Figure 8.2 have to be passed, calibration would be equivalent to analyzing $13 \times 4 \times 15 = 780$ scenarios for each cost-complexity and BNS pruning (for fixed $\bar{n}$ and $\bar{p}$), making it 1560 scenarios altogether. Calibration of $\bar{n}$ and $\bar{p}$ by setting up a grid containing at least five points for $\bar{n}$ (say, 3%, 5%, 10%, 15%, and 20% of the learning sample size) and at least three points for $\bar{p}$ (say, 75%, 87.5%, and 100%) would increase the number of scenarios by 15 times up to $13 \times 4 \times 15 \times 5 \times 3 = 11700$ scenarios. While that would not be a problem in an industrial setting, computing power is quite limited in the framework of this study. At the same time, an empirical rule of thumb suggesting to set $\bar{n}$ to 10% of the learning sample size and $\bar{p} = 75\%$ is likely to provide comparable results. The four calibration constraints increase the probability of selecting those specifications that are more likely to show better performance out-of-sample and implicitly assist to control the appropriate values of $\bar{n}$ and $\bar{p}$, although certainly just to a moderate extent.

Stock	Specification	$\bar{R}$	Learning sample
ADS	fund. and tech.	0.5%	sliding window
ALT	fundamental	0.5%	sliding window
ALV	fundamental	0.5%	sliding window
BAS	fund. and tech.	0.5%	expanding
BAY	fundamental	0.5%	sliding window
BMW	N/A	N/A	N/A
DCX	N/A	N/A	N/A
EOA	N/A	N/A	N/A
LHA	N/A	N/A	N/A
LIN	N/A	N/A	N/A
MAN	N/A	N/A	N/A
SAP	N/A	N/A	N/A
SCH	fundamental	0.5%	expanding
SIE	N/A	N/A	N/A
TUI	N/A	N/A	N/A

Table 8.5: Calibration results for cost-complexity tree pruning, N/A indicates situations when none of the inputs were able to produce a positive calibration yield

Such calibration was performed independently for the BNS and cost-complexity approaches of tree pruning. Tenfold cross-validation and 1-SE rule were employed to find optimal cost-complexity trees. In case when the resulting optimal tree was obviously underparameterized (consisted of the single root node after pruning), the 0-SE rule was employed instead, see Section 5.3 for formal definitions. Table 8.3 and Table 8.5 pro-

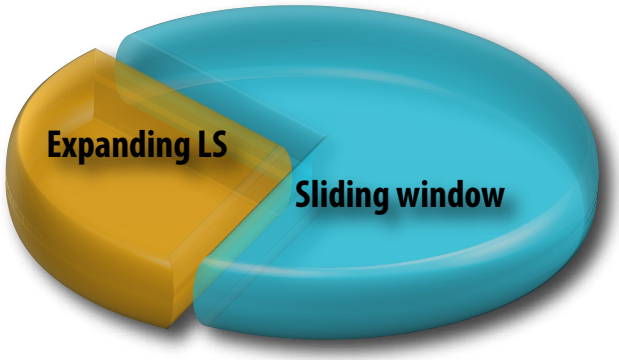

Figure 8.3: Overall calibration results: learning sample type distribution

vide calibration results. Note that in each case that passed the calibration phase, the obtained values of $\bar{R}$ are nonzero suggesting that the 'big hit' ability may be quite an important feature of this and other similar applications where various predictions of the same class may have different economic values. $\bar{R} = 0.5\%$ is obtained in 60% of cases for BNS and in 100% of cases for cost-complexity pruning, or in 75% of cases overall. Values of $\bar{R}$ greater than 1% never appeared to be the optimal choices, which may partly be limited by the four calibration constraints among other factors.

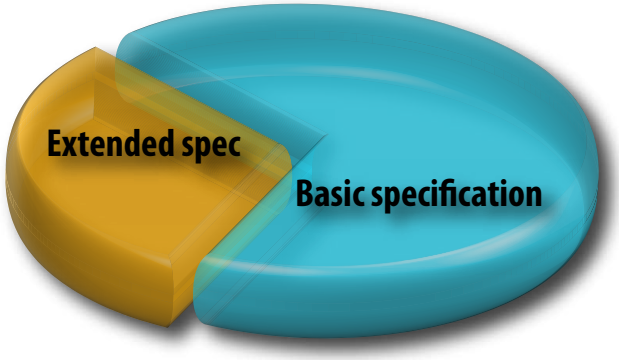

Figure 8.4: Overall calibration results: input specification type distribution

Sliding window appeared to be a more frequent choice for BNS (70% of cases) and cost-complexity pruning (67% of cases), or in 69% of cases overall, than a learning sample with the dynamically increasing size. However, one should recall that sliding window has a higher priority during calibration (refer to Figure 8.2).

The basic feature set was preferred in 70% of cases for BNS and in 67% of cases for cost-complexity pruning, or in 69% of cases overall, but one should again recall the lower calibration priority of the extended feature set.

Overall, roughly in 75% of cases, calibration suggested to employ the class assignment rule equivalent to setting $\bar{R} = 0.5\%$, sliding window learning sample type, and basic feature set. However, the distribution of $\bar{R}$ varies a lot among the two pruning techniques employed.

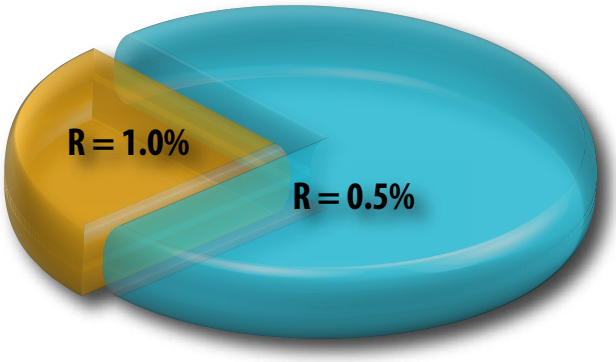

Figure 8.5: Overall calibration results: a distribution of the class assignment rule threshold values $\bar{R}$

As it can be seen from Table 8.3 and Table 8.5, 10 out of 15 stocks (67%) showed positive performance at the calibration set when BNS was employed and only 6 out of 15 (40%) – for cost-complexity tree pruning, which could be an indicator of some inevitable rigidity that the cost-complexity algorithm possesses due to the symmetrical way of pruning.

8.5 Trading Strategies Backtesting

After the class assignment parameter $\bar{R}$, input variable specification, and type of the learning sample are calibrated for each stock independently for BNS and cost-complexity pruning, the next step is to backtest the financial performance of the trading strategies based on the traditional and novel ways of tree optimization. This can be achieved in several ways, and to ensure the unbiased comparison of trading algorithms, no additional sophisticated risk-return portfolio optimizations like finding the minimum-variance frontier, etc. were performed, see Steinbach (2001) for more details on multiperiod mean-variance problems. Instead, two equally weighted portfolios of stocks were created for both BNS and cost-complexity pruning. Portfolio weights were revised each time after new market data became available and new classification trees were constructed:

$$\left[\begin{array}{l} \omega_t = \frac{1}{A_t},\ A_t = \{\#\hat{y}_t : \hat{y}_t \in \{long, short\}\} \text{ if } A_t > 0, \\ \omega_t = 0 \text{ if } A_t = 0 \end{array}\right. \tag{8.3}$$

where $\{\omega_t, \ldots, \omega_t\}$ are portfolio weights at the time period t, $\hat{y}_t$ – forecasted trading position at time t, A_t – the number of active positions at time t. These values are computed independently for portfolios produced by trees optimized via cost-complexity and BNS pruning.

Equally weighted portfolios are a common setting in stock picking applications when various trading strategies are to be empirically compared (Amenc et al., 2003; Brennan et al., 1999; Sorensen et al., 1999; Seshadri, 2003). It is true for any $A_t > 0$ that the expected return of an equally weighted portfolio is the average value of the stocks, and the variance of an equally weighted portfolio is equal to its average covariance, including variances, see Markowitz (1991) for more details.

As it is done traditionally, active positions were closed at the end of each period and no reinvestment was allowed. Each transaction was subject to transactions costs in the amount of 10 b.p., which is a reasonable amount for the institutional investors. According to Table 8.3 and Table 8.5, the first portfolio to backtest contained 10 stocks (BNS pruning) while the second one – 6 stocks (cost-complexity pruning).

Figure 8.6 depicts portfolio's weekly returns when BNS was used for tree pruning. Its annualized return is 17.17% while the Sharpe ratio is 1.26 for the risk-free rate of 4.5%. The Sharpe information ratio (Sharpe, 1994) is one of the standard ways to assess investment strategies in terms of both return and risk. The Sharpe ratio indicates the historic average (or expected) differential return per unit of historic (or expected) variability of the differential return. This ratio of expected added return per unit of added risk provides a convenient measure of the difference between the return of a fund and that of a relevant benchmark, which is commonly set to the risk-free rate R_f:

$$SR = \frac{\bar{\Pi}_{\text{ann}} - R_f}{\sigma_{\text{ann}}(\Pi)} \tag{8.4}$$

where the annualized average portfolio return $\bar{\Pi}_{\text{ann}}$ and standard deviation $\sigma_{\text{ann}}(\Pi)$ are computed as follows for weekly data:

$$\bar{\Pi}_{\text{ann}} = \frac{1}{T}\sum_{i=1}^{T}\Pi_t \times 52, \tag{8.5}$$

$$\sigma_{\text{ann}}(\Pi) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\Pi_t - \bar{\Pi}_{\text{ann}}\right)^2} \times \sqrt{52}. \tag{8.6}$$

The hit ratio of BNS pruning portfolio is 59%. However, one may notice that the vast majority of wrong classifications coincides with the relatively small values of stock price returns, which may be an indicator of the successfully captured 'big hit' ability described in Section 8.2. Overall, this results in a substantial profit at the end of the backtesting period and high Sharpe ratio suggesting the comparatively low return volatility given the excess return.

While the hit ratio of the second portfolio, which was built by employing the traditional cost-complexity approach, is close to the first one – 54%, the financial performance is far more different, refer to Figure 8.7 for details. Although it manages to produce the positive annualized profit – 2.87%, its returns are obviously more volatile resulting in the Sharpe ratio of only -0.09.

BNS exhibited superior performance when compared to the cost-complexity approach, however, another indirect comparison is also possible. In Chapter 1 several empirical studies that employ decision trees for asset allocation are mentioned. Although the markets and time periods are different from those in this work, it may still be of interest to compare these results in terms of relative returns and risk. In Seshadri (2003) the three-class (*overweight*, *underweight*, *neutral*) recommendations for stocks from the S&P500 universe were provided. As at August 6, 2003, the model has returned 14.6% (annualized) with a corresponding Sharpe ratio of 1.5.

Similarly, technological stocks were classified into three performance buckets in Sorensen et al. (1999), and for the period of 1996-1999 the model returned 19.62%

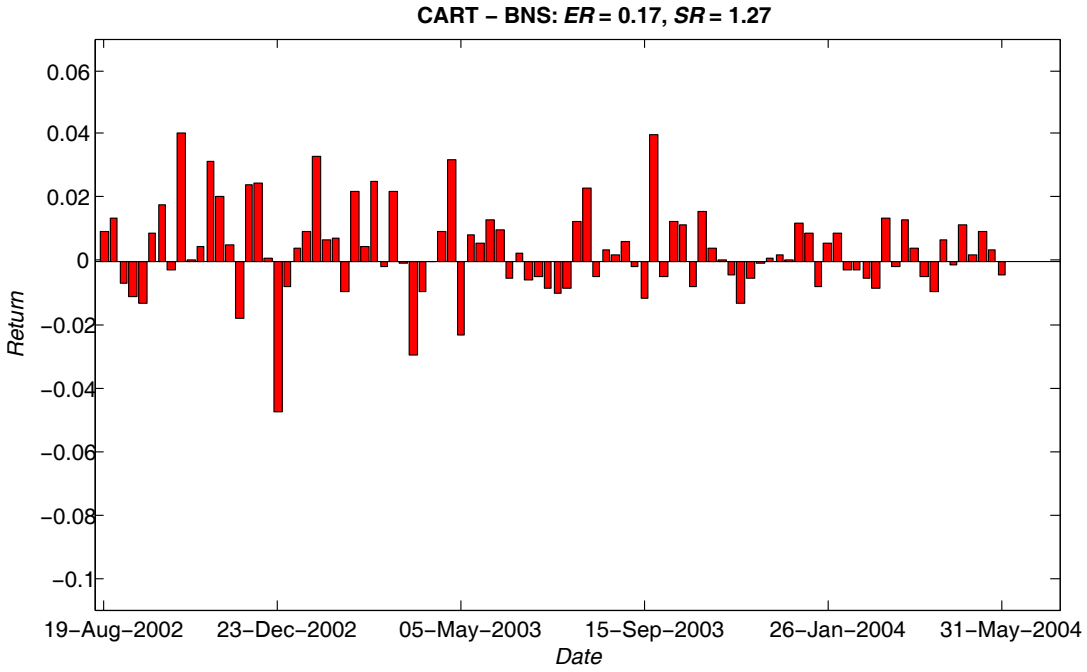

Figure 8.6: Equally weighted portfolio of stocks performance when BNS is employed for tree pruning, ER – the annualized expected return, SR – the Sharpe ratio

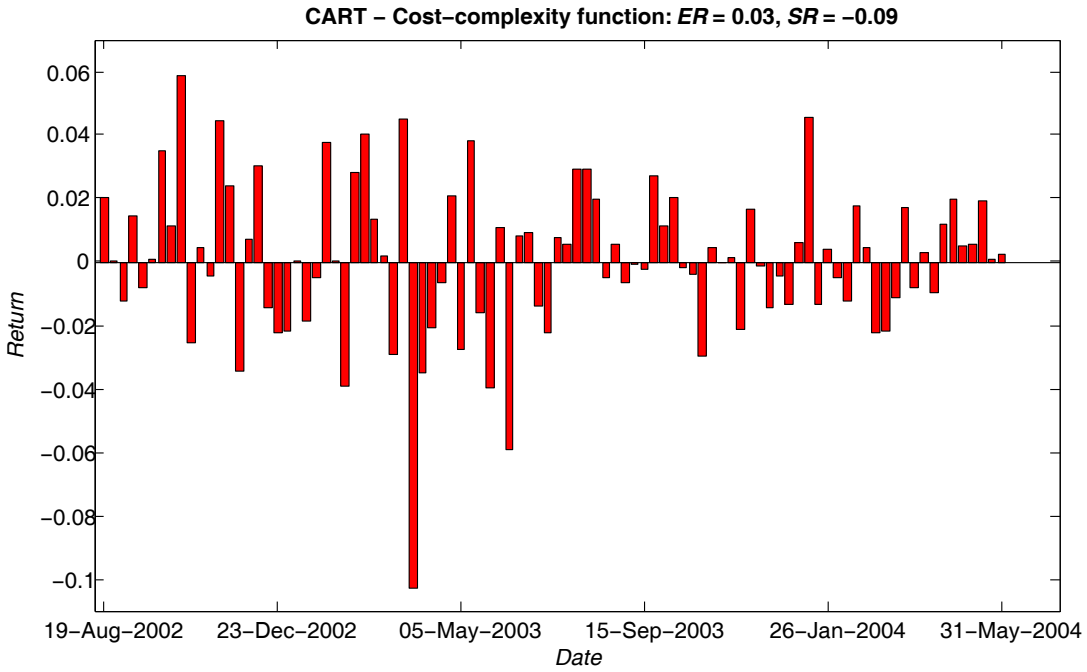

Figure 8.7: Equally weighted portfolio of stocks performance when the traditional cost-complexity approach is employed for tree pruning, ER – the annualized expected return, SR – the Sharpe ratio

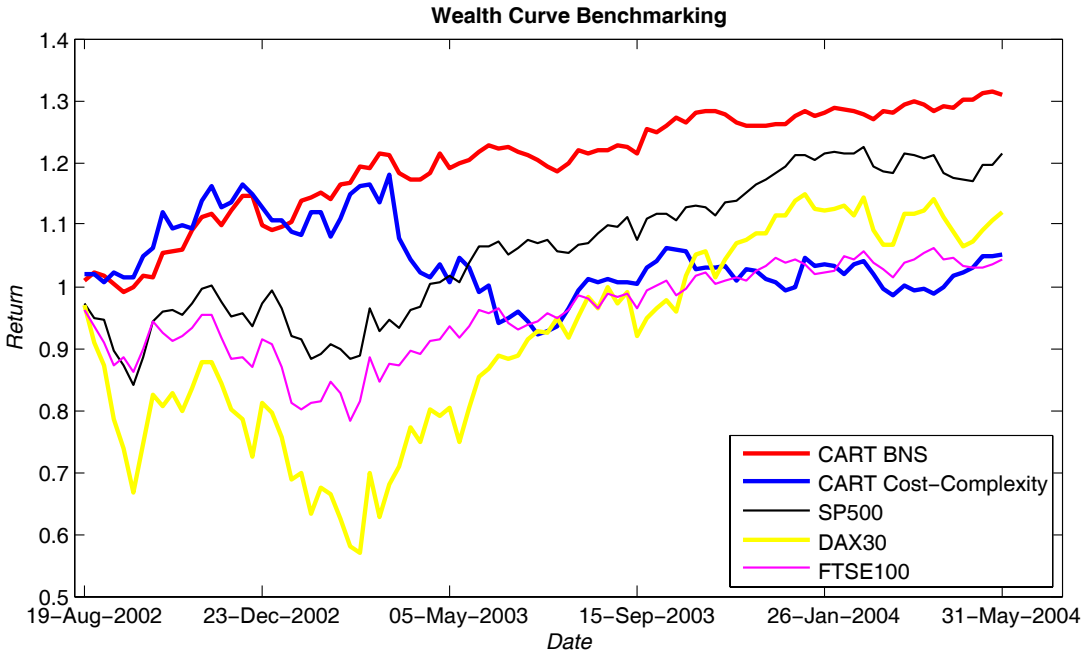

Figure 8.8: Wealth curves for two active decision tree forecasting strategies and three passive investment strategies

with a corresponding Sharpe ratio of 1.23. Interestingly, while the recursive partitioning mechanism is described in this report, nothing is said about tree pruning that led to the achieved performance.

Figure 8.8 depicts the benchmarking of different strategies when an investor has alternative opportunities to invest into DAX30, FTSE100, or SP500 index funds providing a direct comparison with popular alternative passive investment strategies for institutional investors.

Figure 8.9 combines both trees (picked at a randomly chosen time moment) obtained via cost-complexity and BNS pruning for illustrative purposes. The traditionally pruned tree has three terminal nodes, and the resulting decision rule is rather simple. BNS employs the structure of the maximum tree deeper, which results into locating reliable terminal nodes further from the tree origin. There are three terminal nodes as well, however it is worth pointing out that in all three cases pruning is effectively nonsymmetrical since a corresponding reject area is always present. As one can notice from node homogeneity characteristics, the underlying learning sample does not contain too much noise in this case, which resulted in not drastically different decision rules for BNS and cost-complexity pruning.

8.6 Statistical Significance of the Results

The empirical results presented in Section 8.5 reveal two following patterns:

- hit ratios of both trading strategies (cost-complexity and BNS pruning) are very similar,

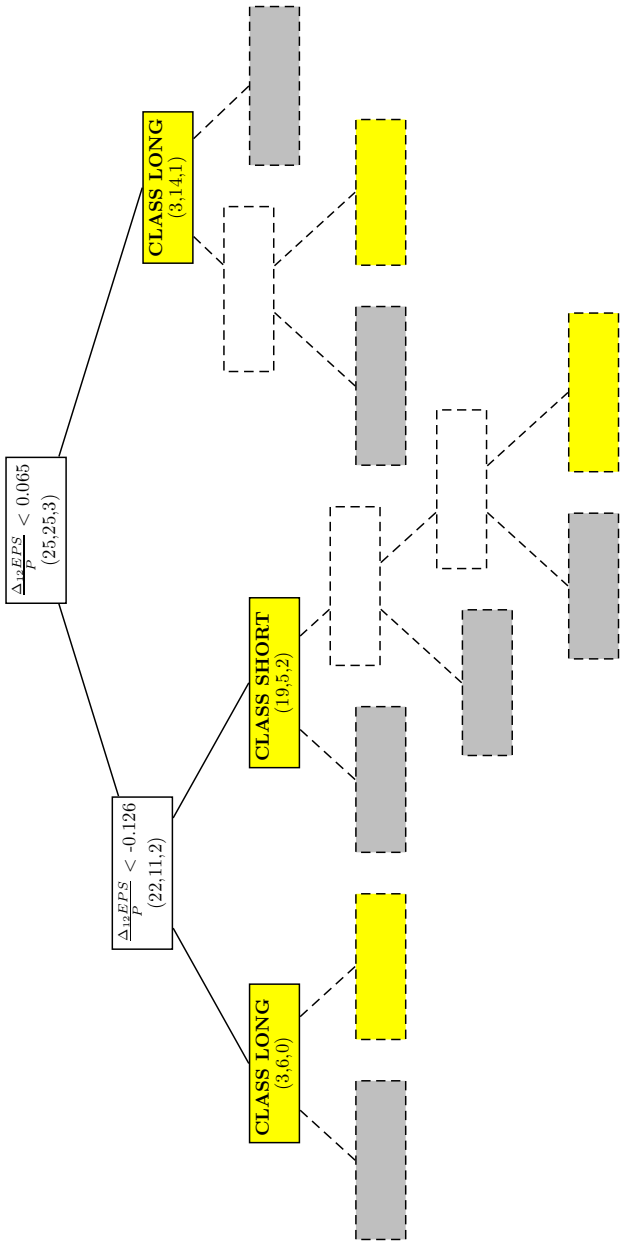

Figure 8.9: Cross-comparison of cost-complexity and BNS pruning: ADS stock

- risk-return characteristics differ quite a lot.

Given the similar hit ratios, the next natural step is to analyze whether the overall financial performance is significantly different for competing forecasting algorithms or the obtained differences in risk-return characteristics are chance results implying that the overall performance difference is spurious. To answer this question, the Diebold-Mariano test was employed (Diebold and Mariano, 1995).

While the hit ratios of the compared portfolios are quite close (54% and 59%), the main motivation to use this test is to take into account the economic value of the

forecasts and not just their directional accuracy. Put differently, the Diebold-Mariano test is capable to determine whether one forecasting method overperforms another due to the 'big hit' ability when hit ratios are very close. The null hypothesis H_0 of the Diebold-Mariano test is that the expected value of an arbitrary loss differential d is equal to zero:

$$H_0 : \mathbb{E}(d) \equiv \mathbb{E}\left[g\left(e^{BNS}\right) - g\left(e^{CC}\right)\right] = 0 \tag{8.7}$$

where $g(\cdot)$ is an arbitrary function and e^{BNS}, e^{CC} – vectors of forecast errors associated with the BNS and cost-complexity portfolios.

Since the Diebold-Mariano test is employed here to compare the expected economic values of two forecasts, the function $g(\cdot)$ resembles the wealth curves from Figure 8.8:

$$\begin{cases} g(e_1) = 1 + e_1, \\ g(e_t) = g(e_{t-1}) + e_t, \quad 1 < t \leq T \end{cases} \tag{8.8}$$

where e_1 is the forecast error at the first time period, T – number of forecasts made (the length of the backtesting period).

Forecast errors are computed as the difference between the realized portfolio profit and any arbitrary benchmark – the resulting form of the loss differential d is invariant with respect to the choice of the benchmark as it is shown below. Given the equation (8.8), if Π^{BNS} and Π^{CC} are vectors of values of the two respective wealth curves and Π^{DAX} is the vector of values of some arbitrary DAX benchmark, then:

$$\mathbb{E}(d) = \mathbb{E}\left[\left(\Pi^{BNS} - \Pi^{DAX}\right) - \left(\Pi^{CC} - \Pi^{DAX}\right)\right] = \mathbb{E}\left[\Pi^{BNS} - \Pi^{CC}\right], \tag{8.9}$$

and therefore the loss differential d is the difference between wealth curves for BNS and cost-complexity portfolios.

Because in large samples the sample mean loss differential $\bar{d}$ is approximately normally distributed with the mean μ and variance $2\pi\hat{f}_d(0)/T$ (Diebold and Mariano, 1995), the test statistic is defined as

$$DM = \frac{\bar{d}}{\sqrt{2\pi\hat{f}_d(0)/T}} \tag{8.10}$$

where $\bar{d}$ is the sample mean of the the loss differential d, μ is the population mean loss differential, $\hat{f}_d(0)$ is a consistent estimate of spectral density of the loss differential at the zero frequency, and T is the number of forecasts.

The variance $2\pi\hat{f}_d(0)$ was estimated using the Bartlett kernel with automatic bandwidth selection (Andrews, 1991; Newey and West, 1994). As a result, $DM = 13.14$ and the p-value $= 1.37 \cdot 10^{-38}$, which indicates that H_0 is rejected at the 0.1% confidence level. One may therefore conclude that the economic value associated with portfolio returns generated by BNS and cost-complexity decision tree pruning strategies is significantly different in favor of BNS at any reasonable confidence level.

Because calibration was performed identically for BNS and cost-complexity pruning and equally weighted portfolios were created representing the average stock performance in both cases, it follows that at least for the analyzed DAX30 data set, all other things equal, BNS significantly outperformed cost-complexity pruning in economic terms and was approximately on a par when the directional accuracy is assessed.

9 Conclusions

This study provides new empirical evidence on possibilities of successful stock selection for institutional investors via object recognition. While there exist several major ways to model this process, the study carefully examines the most popular of them including general equilibrium models, traditional asset pricing, parametric and semiparametric techniques, and various classification methods. The use of decision trees is justified due to the quite special and advantageous properties of the method, which include excellent interpretability of decision rules, ability to select necessary features of the learning sample automatically, ease of implementation, and computational speed. On this basis, binary classification trees can be concluded to be an excellent modeling choice for the task of stock picking.

Forecasting via decision trees involve several important stages. At first, a maximum tree is to be constructed, and the study meticulously examines various popular tree induction techniques including classical classification tree induction process proposed in Breiman et al. (1987) and popular impurity measures, FACT/QUEST, ID3/C4.5, CHAID, and more sophisticated induction techniques like oblique (OC1) and nonlinear decision trees. These approaches are then critically examined from the perspective of applied environments, and one of the most important conclusions at this stage is the following. Tree induction methods do not exhibit significantly different out-of-sample performance in terms of accuracy when standard pruning is applied. Moreover, even completely random splitting provides comparable results *after* pruning, and the major difference between tree induction methods is a configuration of the rules, which may take sophisticated form to a greater or lower extent depending on a particular method.

The second, and perhaps the crucial, step of applications of decision trees to forecasting is tree pruning – the process of decision rules optimization to avoid over- and underfitting. Generally, pruning methods can be divided into two groups: top-down and bottom-up techniques. Top-down approaches include various early-stopping rules like critical-value pruning and others. Although this group of methods does not require to perform validation of the rules, the cost is usually a severe bias towards underfitting of the final classification trees. Therefore, in this study the emphasis is put on the second group of bottom-up methods that can analyze the complete structure of the maximum tree without the risk of premature pruning as in the case of top-down criteria. Some of top-down methods require to perform validation on a separate test set, which results into sequential comparisons of various tree branches or subtrees like in the case of cost-complexity pruning. However, some other pruning techniques do not necessary require validation. MDL pruning defines the measure of tree quality based on the description length and, starting from the maximum tree, prunes terminal nodes directly – without validation – if the cost of encoding the class labels of observations at the parent node t is lower than or equal to the cost of encoding the subtree rooted at t. Empirical comparisons of various pruning methods indicate very similar performance in terms of the accuracy and more significant differences in the tree size and execution times. On

this basis, cost-complexity pruning was chosen as a standard pruning technique for this study, which is consistent with the choice of default pruning for OC1 tree induction in Murthy et al. (1994).

One of the recent pruning methods introduced in Osei-Bryson (2004) employs multiple quality criteria that evaluate tree performance. This layer of pruning techniques can certainly be of particular interest when user-defined utility preferences need to be taken into account particularly if various classification rules need to be compared.

Ensemble methods are an important tool of machine learning aimed at combining weak classifiers such as decision trees of various configurations in ways that produce an aggregate rule of stronger forecasting power. Many empirical studies show that ensemble methods like bagging or adaptive boosting can be very successful in certain applications. However, produced aggregate rules lack the interpretability of the results, and they are generally significantly slower than single classifiers (especially Adaboost). Because one of the aims of this work is to propose a trading algorithm that is highly interpretable, which is suggested by practical experience of interacting with various financial services organizations, ensemble methods are not considered as proper candidates for stock picking in this study. However, in the realm of numerous hedge funds that employ various aggressive quantitative techniques of arbitrage trading, ensemble methods may become an excellent choice especially in high-frequency trading applications when classification decisions must be carried out automatically and black box models are not a disadvantage. That is why bagging, boosting, and random forests are carefully examined from the practical perspective of stock picking, too.

The aforementioned pruning techniques are symmetrical in the sense that each time two terminal nodes can be either pruned or left in the tree. The novel pruning strategy introduced in this study – Best Node Selection – allows a more flexible approach and can prune only one of the child nodes if necessary. Pruning both child nodes is also an option, and the optimal decision is carried out each time automatically. Apart from that, BNS introduces reject option, which is an option not to classify an observation if the estimated risk of misclassification is high. In contrast to standard pruning methods that operate with node triplets and produce integral (sub-)tree quality indices, BNS operates with each node individually. This provides an opportunity to leave those parts of the tree that contain only few *reliable* nodes and would be pruned traditionally, however the quality of the rule is stabilized via setting reject regions at other *unreliable* nodes of the inevitably kept parts of the rule. Node reliability is controlled by user-defined purity and representativity ratios. Similar to the 1-SE rule in Breiman et al. (1987), an empirical rule of thumb is proposed for automatic selection of the two BNS parameters. However, in industrial settings, a more rigorous approach is also feasible where both parameters can be calibrated given the appropriate quality measure (such as out-of-sample accuracy rate, financial return of classification decisions, etc.).

One of the notorious advantages of BNS pruning is that the method does not require additional validation, which results in excellent computation time. This is a result of a specific BNS *inverse propagation* property that ensures that if a child node is unreliable, its parent node is unreliable, too (therefore eliminating the need of sequential bottom-up pruning) if the size of the tree and impurity measure is set accordingly. A rigorous proof of this fact is provided. Essentially, the application of BNS is equivalent to growing a tree of a specific size, which is a known function of two BNS parameters, and reject regions are computed automatically.

The flexibility of BNS and introduction of reject option come at the cost of two quality parameters that need to be either calibrated or set up according to an empirical rule of thumb. Cost-complexity pruning, for instance, is fully nonparametric, however its architecture is more rigid, which is the inevitable tradeoff. Some applications may require the classification of cases unconditionally (put differently, when reject option is not permitted), and in this case traditional pruning methods like cost-complexity pruning will probably be more reasonable choices.

In order to compare BNS and cost-complexity pruning empirically, a DAX30 stock market data set was employed to perform historical simulation of trading. At the first stage, calibration of model parameters including class assignment rules, type of feature space specification, and learning sample creation method was done. Potentially allowing for a 'big hit' ability, calibration results finally showed no cases when naive classification rules with only classes *long* and *short* were preferred. Backtesting has shown the superiority of BNS in terms of financial performance of the recursive equally weighted portfolio: the annualized profit of 17.17% vs. 2.87% and the Sharpe ratio of 1.27 vs. -0.09. However, with close hit rates of 54% (cost-complexity pruning) and 59% (BNS pruning), the spuriousness of superior financial results ought to have been tested. Wealth curves of the associated trading strategies were compared employing the Diebold-Mariano test that indicated the statistically significant difference of financial performance at any reasonable confidence level in favor of BNS.

While the hit ratios of the both active strategies are quite close and do not significantly deviate from 50% (54% and 59%), the difference in the economic value of both forecasts is undeniably significant. Here is an opinion of professional equity investment managers from *Schroders* (a global asset management company with €164.4 billion under management as at June 30, 2008) commenting a very similar outcome (in Schroders (2006), the backtested annualized return of a decision tree based trading strategy over the whole period is 12%): *'Although these hit rates do not seem significantly different from 50% (which is indicative of no skill in stock picking), this is very typical in financial applications and it would be rare to observe models with average hit rates in excess of 55%. Indeed, as the chart above illustrates, hit rates even slightly better than 50% can generate strong strategy outperformance in practice. [...] We would conclude from this analysis that the model is very successful at locating the key stock characteristics that identify future relative performance'.*

Another important novel feature of this study is a separate analysis of each stock when performing historical simulation of stock picking. Because DAX30 is comprised of companies with backgrounds of various nature, individual decision trees were supposed to deliver a more flexible approach, which is ultimately supported by backtesting results: for any fixed time moment, decision trees for various stocks in the portfolio do not coincide. To illustrate this statement, at randomly selected time point of the backtesting period (it appeared to be week 18), decision trees for all stocks were recorded. Since the maximum tree coincide for BNS and cost-complexity pruning and BNS trees are likely to have bigger size due to reject regions, trees with BNS pruning are provided as relevant examples (see Appendix B). Root node variables represent the features that lead to the maximum initial class variance reduction and can therefore considered to be the most significant. Figure 9.1 provides this distribution and clearly indicates the diversity of the variables, which may be an empirical justification of the way how learning samples were created.

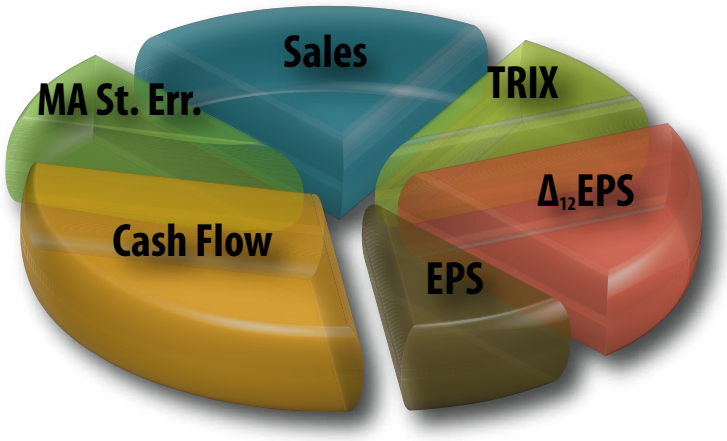

Figure 9.1: Root node variable distribution at week 18 of the validation period (see the decision trees in the Appendix B)

It is worth pointing out that there are certain practical limitations of replicating the described investment strategy for individual investors. First of all, short sales are an option only for institutional investors. Secondly, although direct transactions costs are taken into account, individual investors are likely to bear additional costs for market data acquisition from one of the major providers like *Thomson Datastream* or *Bloomberg*. None of these obstacles exist for institutional investors, amplifying the advantageous features of the trading strategy by the effect of scale.

With the proven reverse propagation property of BNS (see Appendix A), it is easy to build a tree of the optimal size possessing a much more flexible nonsymmetrical structure than its symmetrical canonically pruned counterpart. When reject option is an eligible choice for a classification task, BNS is likely to combine the best features of decision trees including a very high level of interpretability and ease of computations with lucrative properties of reject option that makes the risk of misclassification be manageable by the end-user.

Appendix A

Lemma 1. *Let's t_P be the parent node for t_L and t_R given some arbitrary split s. If the following inequalities hold:*

$$\begin{cases} i(t_P) > i(t_L), \\ i(t_P) \geq i(t_R) \end{cases} \tag{1}$$

and one of them holds as strict, for instance, for t_L, then it is true that

$$\Delta i(t_P, s) = i(t_P) - p_L i(t_L) - p_R i(t_R) > 0. \tag{2}$$

The reverse statement is also true.

Proof. The proof of the first part is straightforward and can be found in Breiman et al. (1987). Let us prove the reverse part of the lemma. Using the link between p_L and p_R, one can get the following inequality:

$$\begin{cases} \Delta i(t_P, s) = i(t_P) - p_L i(t_L) - p_R i(t_R) > 0, \\ p_L + p_R = 1,\ p_L \in (0; 1),\ p_R \in (0; 1) \end{cases}$$

$$\Rightarrow\ i(t_P) > p_L i(t_L) + (1 - p_L) i(t_R) \tag{3}$$

Let us suppose that $i(t_P) < i(t_L)$ and to be more specific: $i(t_P) = p_L i(t_L) < i(t_L)$ $\forall p_L \in (0; 1)$. Then $p_L i(t_L) > p_L i(t_L) + (1 - p_L) i(t_R)$ that is equivalent to $(1 - p_L) i(t_R) < 0 \ \Leftrightarrow\ i(t_R) < 0$, which is impossible by the definition of $i(\cdot)$. Hence one can conclude that $i(t_P) \geq i(t_L)$.

Let us suppose now that $i(t_P) < i(t_R)$ and let $i(t_P) = (1 - p_L) i(t_R) < i(t_R) \quad \forall p_L \in (0; 1)$. Then $(1 - p_L) i(t_R) > p_L i(t_L) + (1 - p_L) i(t_R) \ \Leftrightarrow\ i(t_L) < 0$, which is impossible. That is why $i(t_P) \geq i(t_R)$.

The remaining step is to note that one of the two inequalities – either $i(t_P) \geq i(t_L)$ or $i(t_P) \geq i(t_R)$ – must hold as strict because if $i(t_P) = i(t_L) = i(t_R)$, then $\Delta i(t_P, s) = 0$ that violates the conditions of the lemma. □

Lemma 2. *Let t_L and t_R be the two child nodes with t_P being the parent node and s – the relevant data split so that $\Delta i(t_P, s) > 0$. Let $S(t)$ be the dominating class of the node t. Then for the node $t \in \{t_L, t_R\}$ so that $S(t) = S(t_P)$ it is true that if $v(t) = 0$, then $v(t_P) = 0$ where $v(\cdot)$ is defined in (7.1) so that $\bar{n}$ and $\bar{p}$ do not violate (7.2).*

Proof.

1. Let us consider two sets of conditional probabilities $(p_1, p_2, \ldots, p_J)$ and $(p'_1, p'_2, \ldots,$ $\ldots, p'_J)$ where $p_i = p(i|\, t_P)$ and $p'_i = p(i|\, t)$, $t \in \{t_L, t_R\}$. Since the inequality $i(t_P) > i(t)$ holds as strict at least for one of the child nodes $\{t_L, t_R\}$ (Lemma 1), it follows that at least one of the values in the set $p(\, i|\, t)$ has changed compared to the set $p(\, i|\, t_P)$, refer to Breiman et al. (1987) for the detailed description of the properties of an arbitrary impurity function.

2. Since $\sum\limits_{i=1}^{J} p(\, i|\, t) = 1$, there exist at least one value of the conditional probability from $(p'_1, p'_2, \ldots, p'_J)$ that has *increased* compared to $(p_1, p_2, \ldots, p_J)$ and at least one that has *decreased* because the situation when each of the components $p(\, i|\, t) \geq 0$ changed their values in one direction is impossible.

3. For some class j, let $p'_j = \max\limits_i p'_i = \max\limits_i p(\, i|\, t)$, i.e. the maximum value of the conditional probability from the second set. Then while there may exist an arbitrary number of components that increased or decreased their values when transferring from the first set of probabilities $p(\, i|\, t_P)$ to the second – $p(\, i|\, t)$, p'_j is the maximum value from the subset of values that have *increased*.

4. That is why $p_j \leq p'_j$ where $p_j = \max\limits_i p(\, i|\, t)$ and $p'_j = p(j|t_P)$.

   Since $j = \text{argmax}_i\, p\,(i\,|t)$, it follows that $S(t_P) = j$. It is given that $S(t) = S(t_P)$, therefore $S(t) = j$. Because $v(t) = 0$, it follows that $p(j|t) < \bar{p}$. However, it was proven that $p(j|t) \geq p(j|t_P)$. Therefore, $p(j|t_P) \leq p(j|t) < \bar{p}$. Hence $p(j|t_P) < \bar{p} \;\Rightarrow v(t_P) = 0$.

□

Theorem 1. *Let t_L and t_R be the two child nodes with t_P being the parent node and s – the relevant data split. Let t_L and t_R be terminal nodes in a tree $T(\frac{\bar{n}}{\bar{p}})$. Let $i(t)$ be the impurity function taking the form of the Gini index: $i(t) = 1 - \sum\limits_{j=1}^{J} p^2(j|t)$, J be the number of classes in the learning sample and $\Delta i(t_P, s) > 0$. Then if at least one of the child nodes is unreliable: $v(t) = 0$, then the parent node is also unreliable: $v(t_P) = 0$ where $v(\cdot)$ is defined in (7.1) so that $\bar{n}$ and $\bar{p}$ do not violate (7.2).*

Proof. Let $j^* = \text{argmax}_i\, p\,(i\,|t)$. One of the requirements for a node to be accounted as reliable is to show the significantly high probability of the dominating class: $p(j^*|t) \geq \bar{p}$. Since $\sum\limits_{i=1}^{J} p(\, i|\, t) = 1$, $0 \leq p(\, i|\, t) \leq 1$, and $i(t) = 1 - \sum\limits_{j=1}^{J} p^2(j|t)$, then the inequality $p(j^*|t) \geq \bar{p}$ implies the existence of the upper bound of the node impurity value – $\bar{\imath}$, so that

$$p(j^*|t) \geq \bar{p} \Leftrightarrow i(t) \leq \bar{\imath}$$

where

$$\bar{\imath} = \begin{cases} 1 - \frac{\bar{p}^2}{J}, \ \bar{p} = \frac{1}{J} \\ \frac{-J\bar{p}^2+2\bar{p}+J-2}{J-1}, \ \bar{p} > \frac{1}{J} \end{cases}$$

Since $v(t) = 0$, there are two possible configurations of the triplet $\{t_L, t_R, t_P\}$, where t_L and t_R are arbitrary child nodes and t_P – their parent node.

1. Both child nodes are unreliable: $v(t_L) = v(t_R) = 0$

   In this case $i(t) > \bar{\imath}$ where $t = \{t_L, t_R\}$ because $t \in \tilde{T}(\frac{\bar{n}}{\bar{p}})$. Since $\Delta i(t_P, s) > 0$, according to Lemma 1 it follows that $i(t_P) \geq i(t)$, and therefore $i(t_P) > \bar{\imath} \Rightarrow v(t_P) = 0$.

2. Only one of the child nodes is unreliable, for the sake of simplicity let it be the node t_R:

   Employing Lemma 1 once again, it is possible to conclude that $i(t_P) \geq i(t_L)$. Because the node t_L is pure, then $i(t_L) < \bar{\imath}$. However, it is not possible to say if $i(t_P) > \bar{\imath}$ or not.

   But for the node t_R the situation changes drastically. Again, $i(t_P) \geq i(t_R)$, but in this case $i(t_R) > \bar{\imath}$, so one can conclude that $i(t_P) > \bar{\imath} \Rightarrow v(t_P) = 0$.

Since it is given that $v(t) = 0$, the situation when both terminal nodes in the triplet are pure is impossible. This concludes the proof of the theorem.

If t_P is unreliable, the same set of arguments can be applied to this node. Therefore, if a terminal node in $T(\frac{\bar{n}}{\bar{p}})$ is unreliable, each of its parent nodes is unreliable, too. □

Appendix B

This part of the Appendix contains performance charts for all of the stocks in both portfolios (cost-complexity and BNS pruning). Market returns are provided for comparison.

To be able to compare visually the structure of the decision rules and assess the adequacy of the use of individual trees for each of the stocks, a set of decision trees for a randomly selected time point (week 18) of the backtesting period is provided. The distribution of the root node variables, corresponding to these trees, is available in Chapter 9.

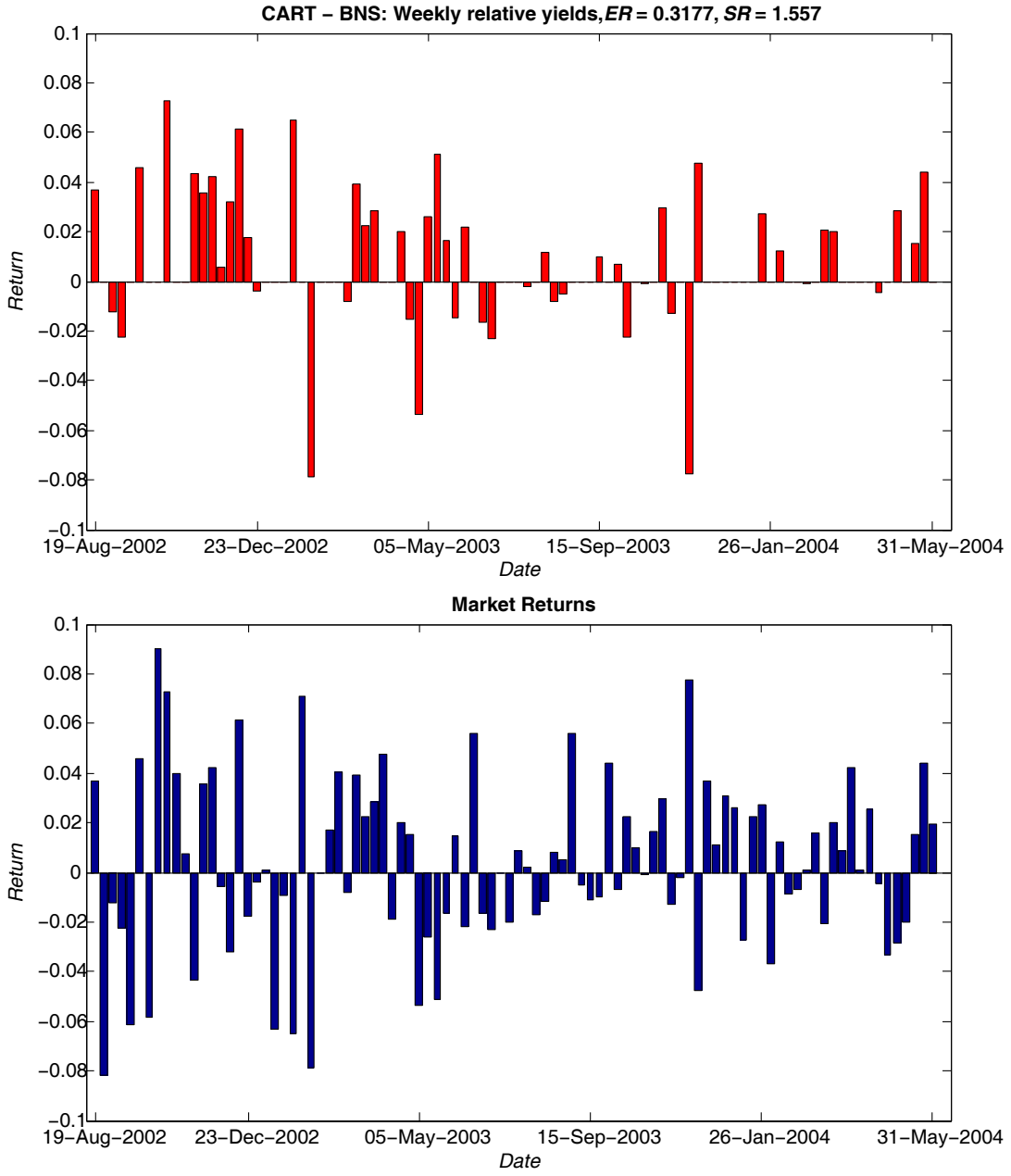

Figure 2: Backtesting performance of the ADS stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

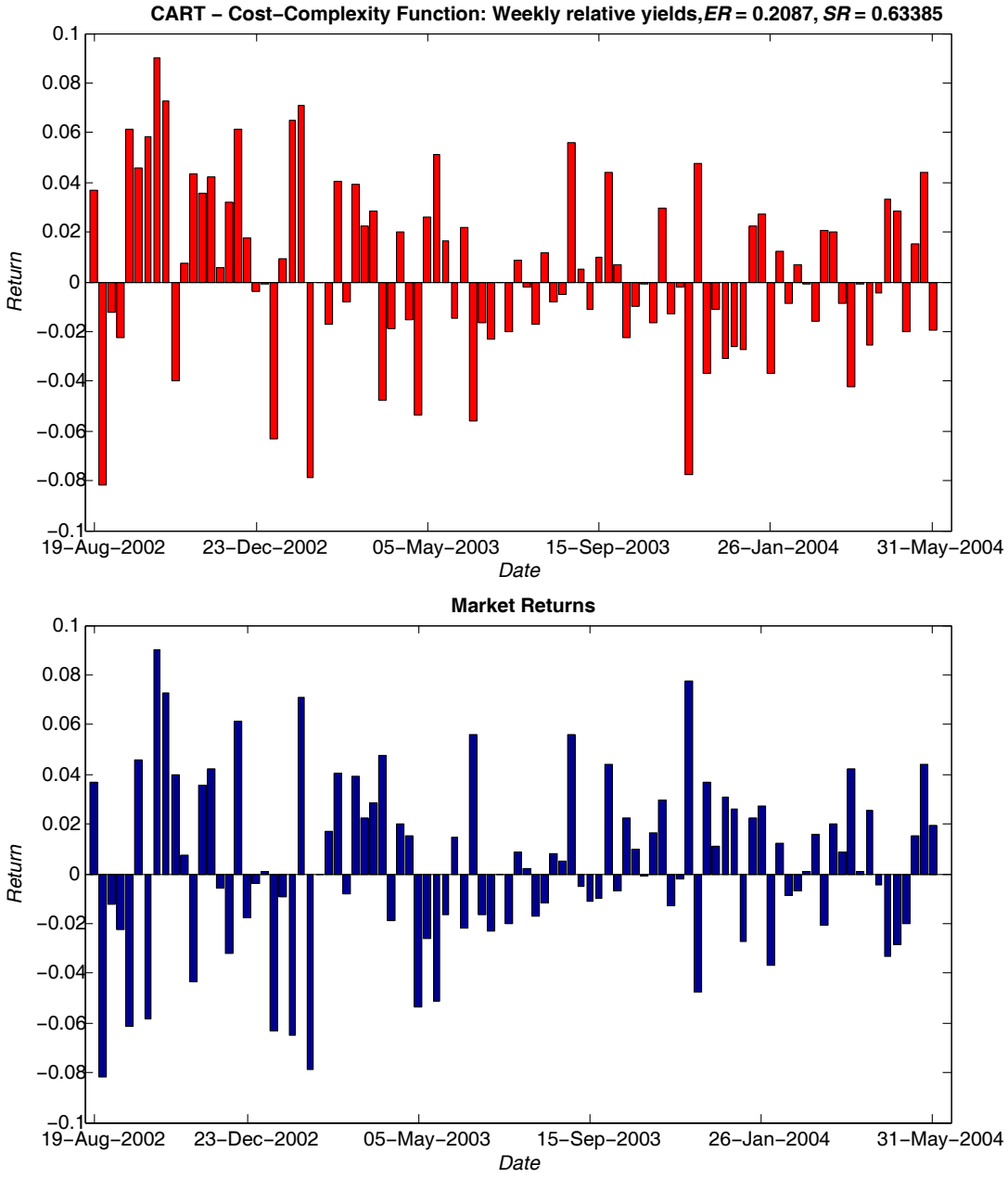

Figure 3: Backtesting performance of the ADS stock when the cost-complexity tradeoff is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

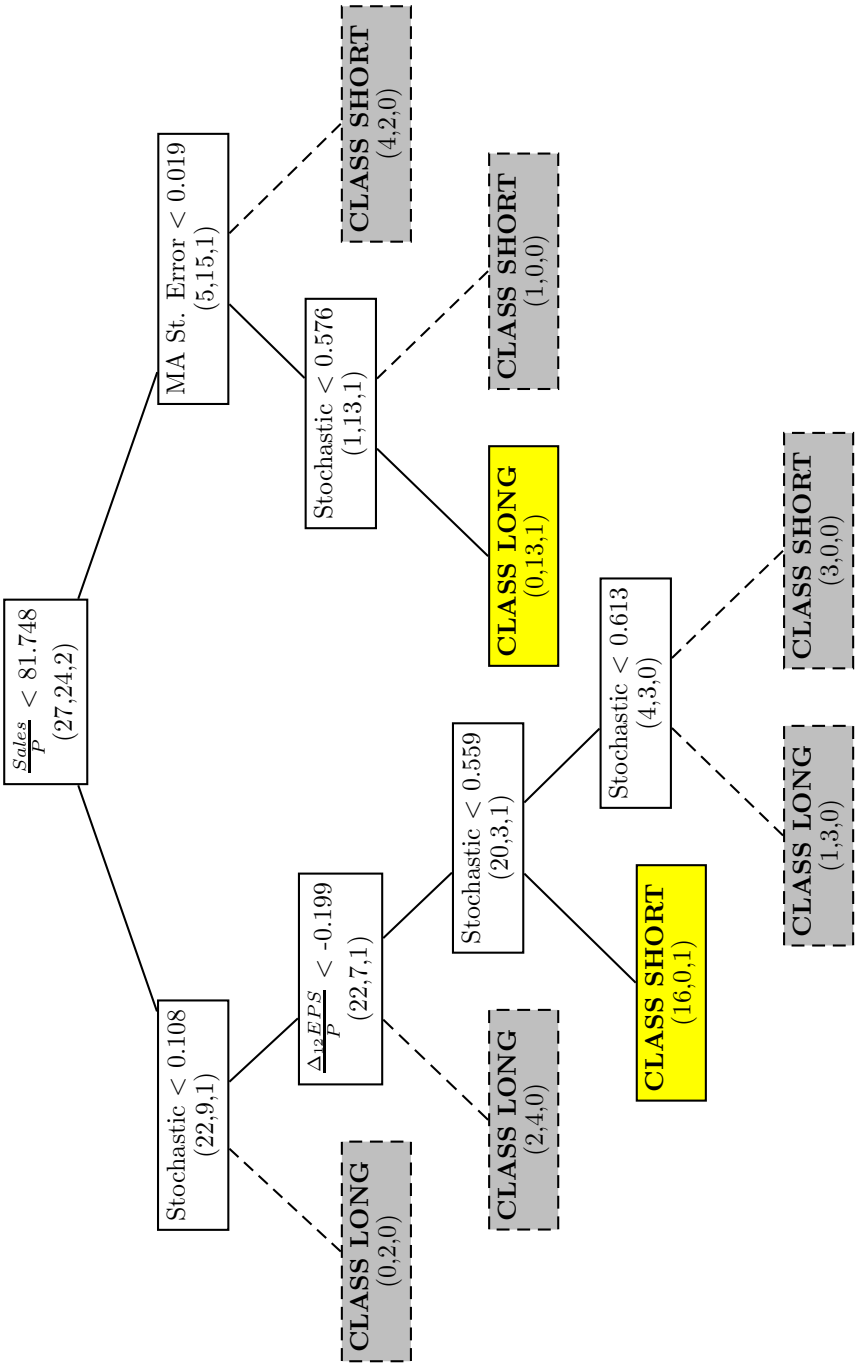

Figure 4: ADS stock classification tree, BNS pruning

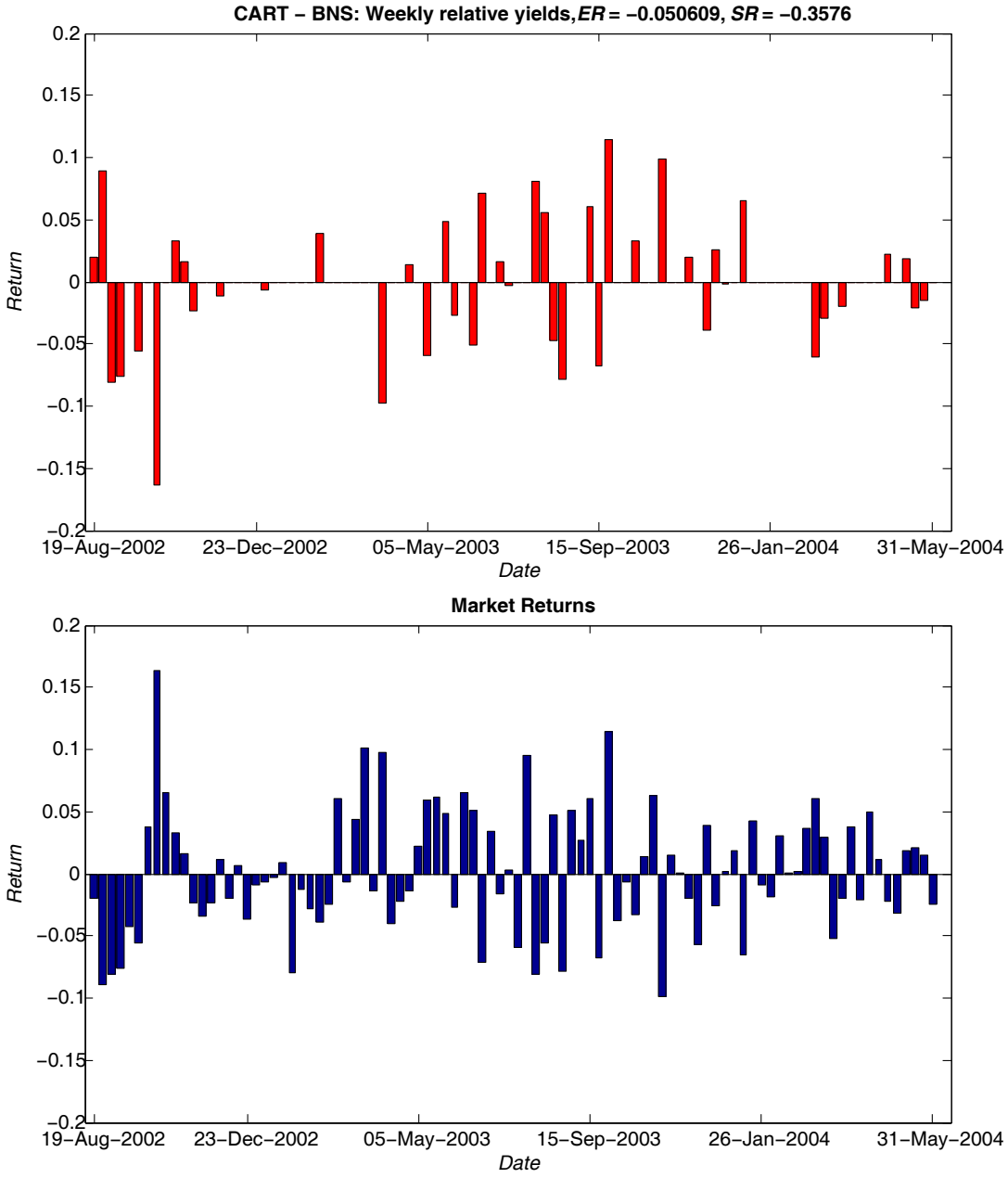

Figure 5: Backtesting performance of the ALT stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

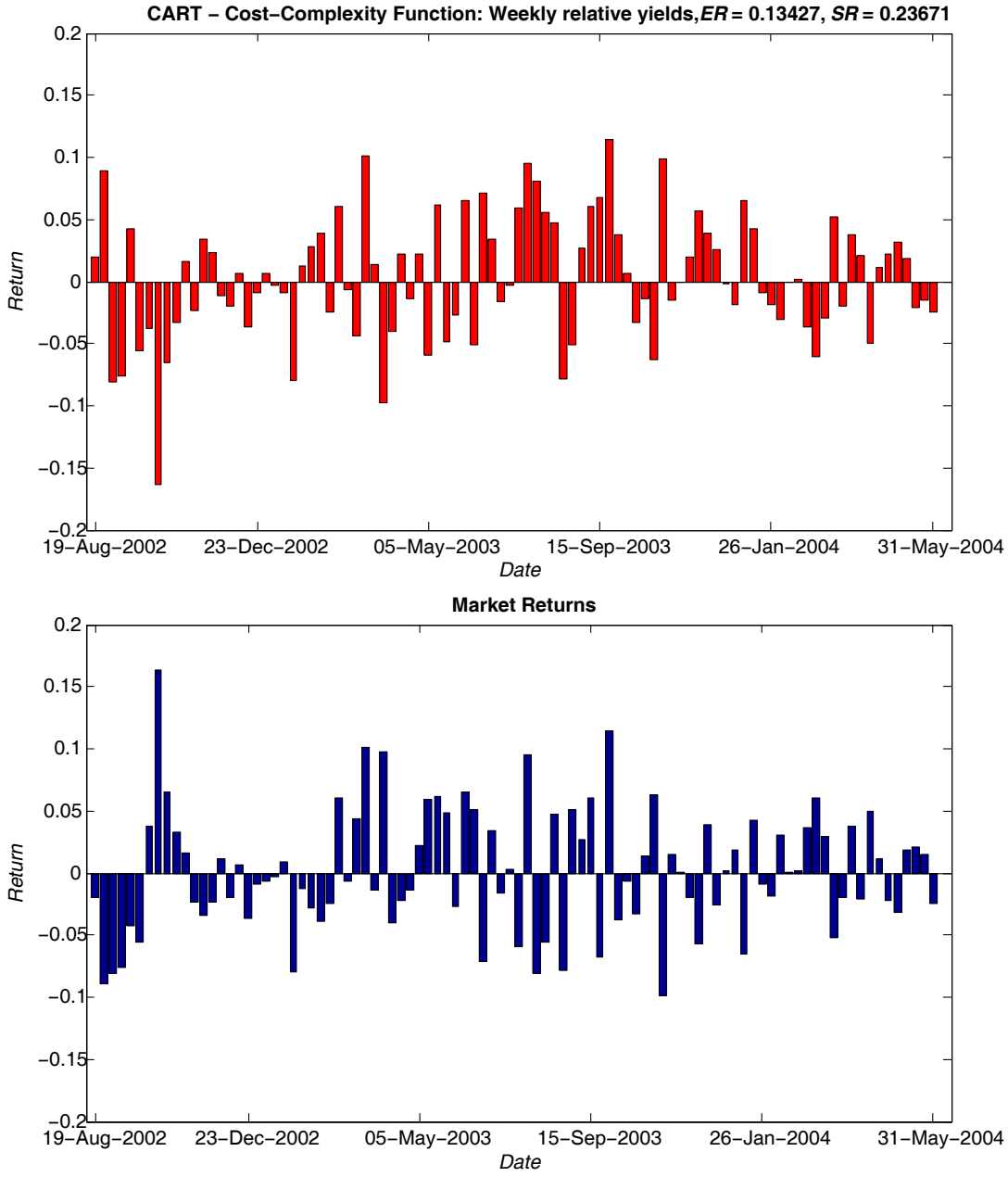

Figure 6: Backtesting performance of the ALT stock when the cost-complexity tradeoff is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

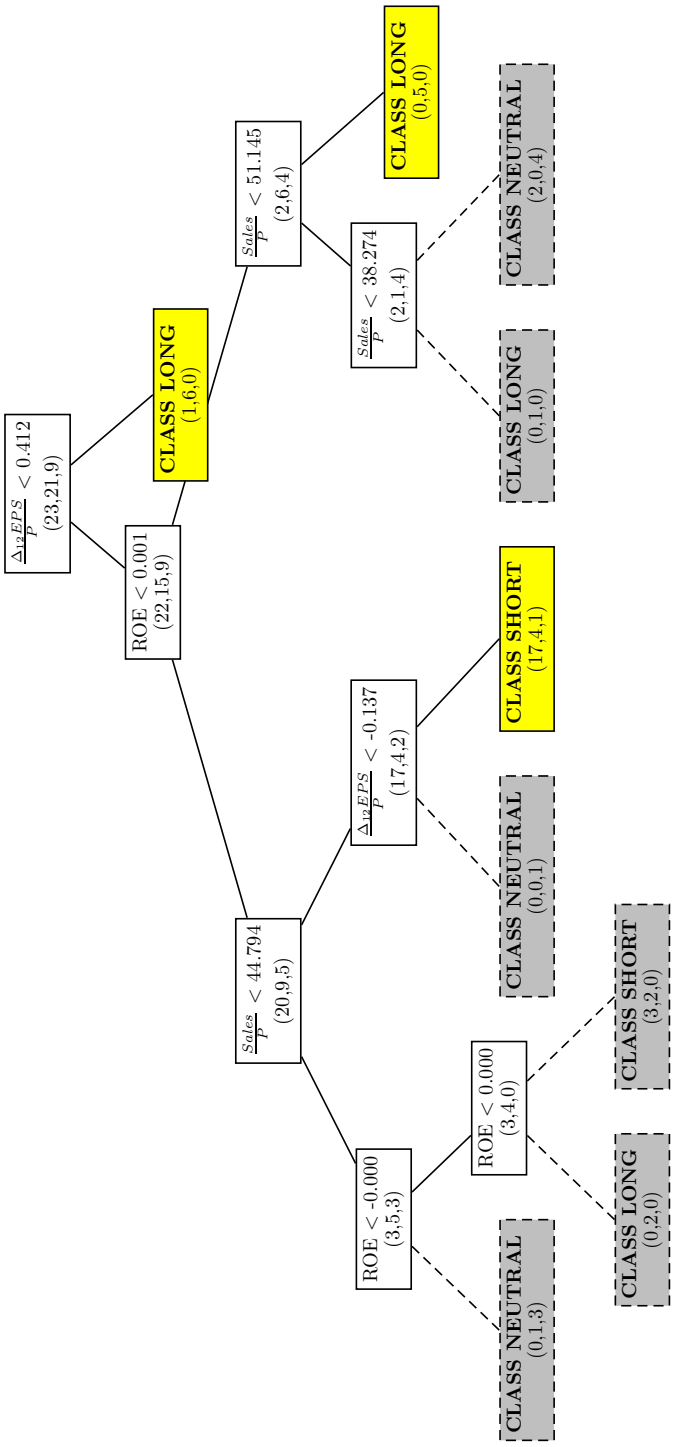

Figure 7: ALT stock classification tree, BNS pruning

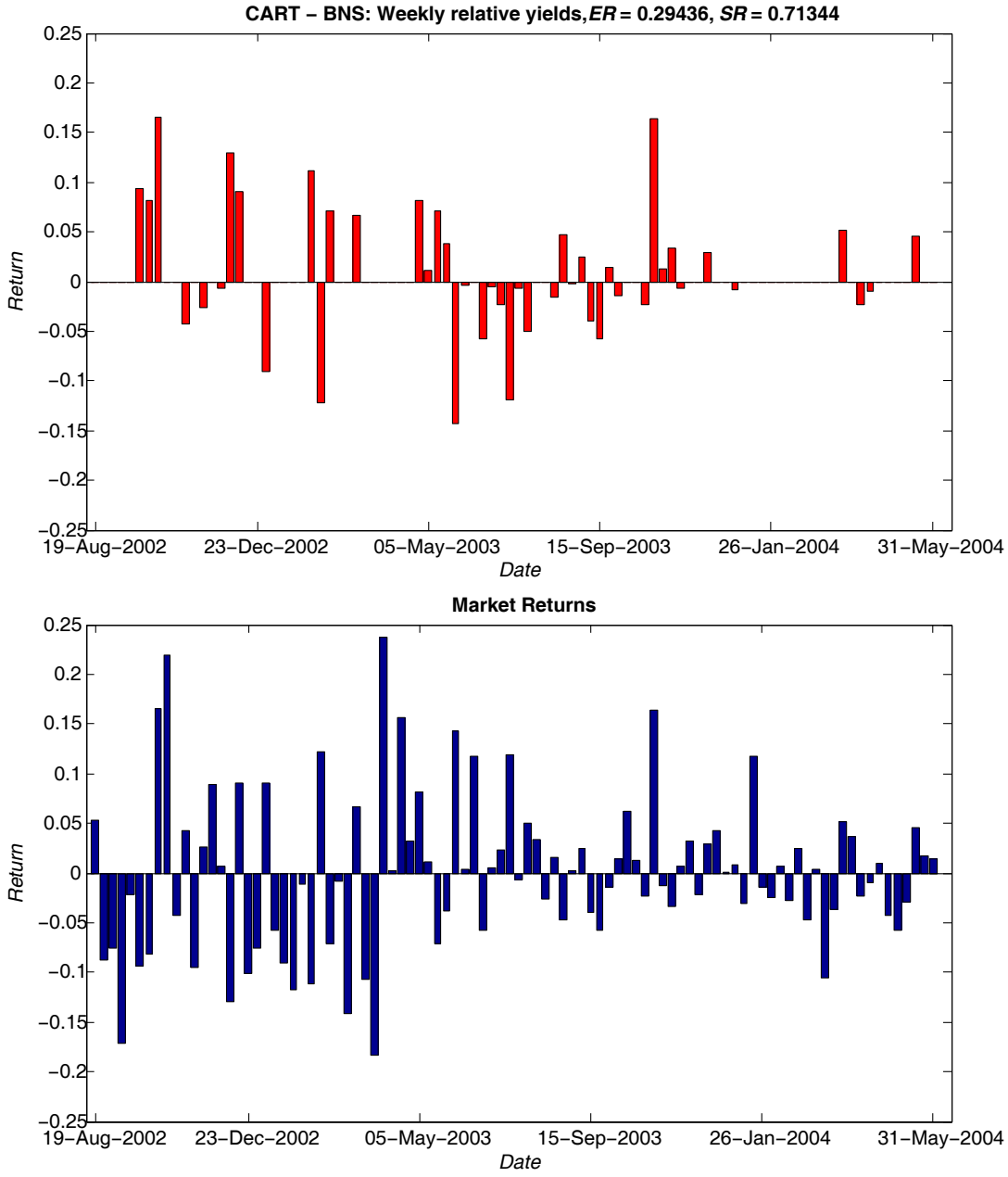

Figure 8: Backtesting performance of the ALV stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

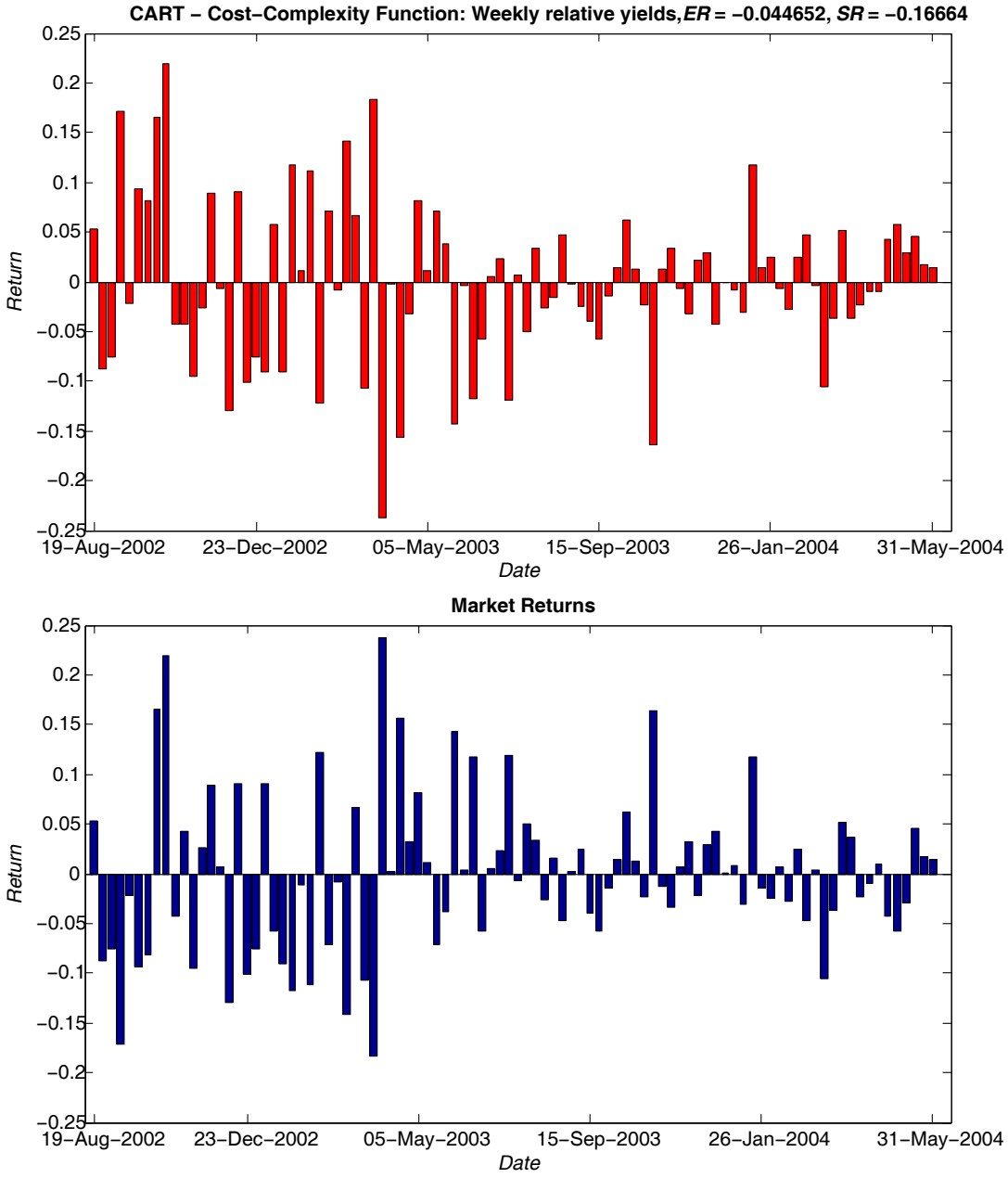

Figure 9: Backtesting performance of the ALV stock when the cost-complexity tradeoff is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

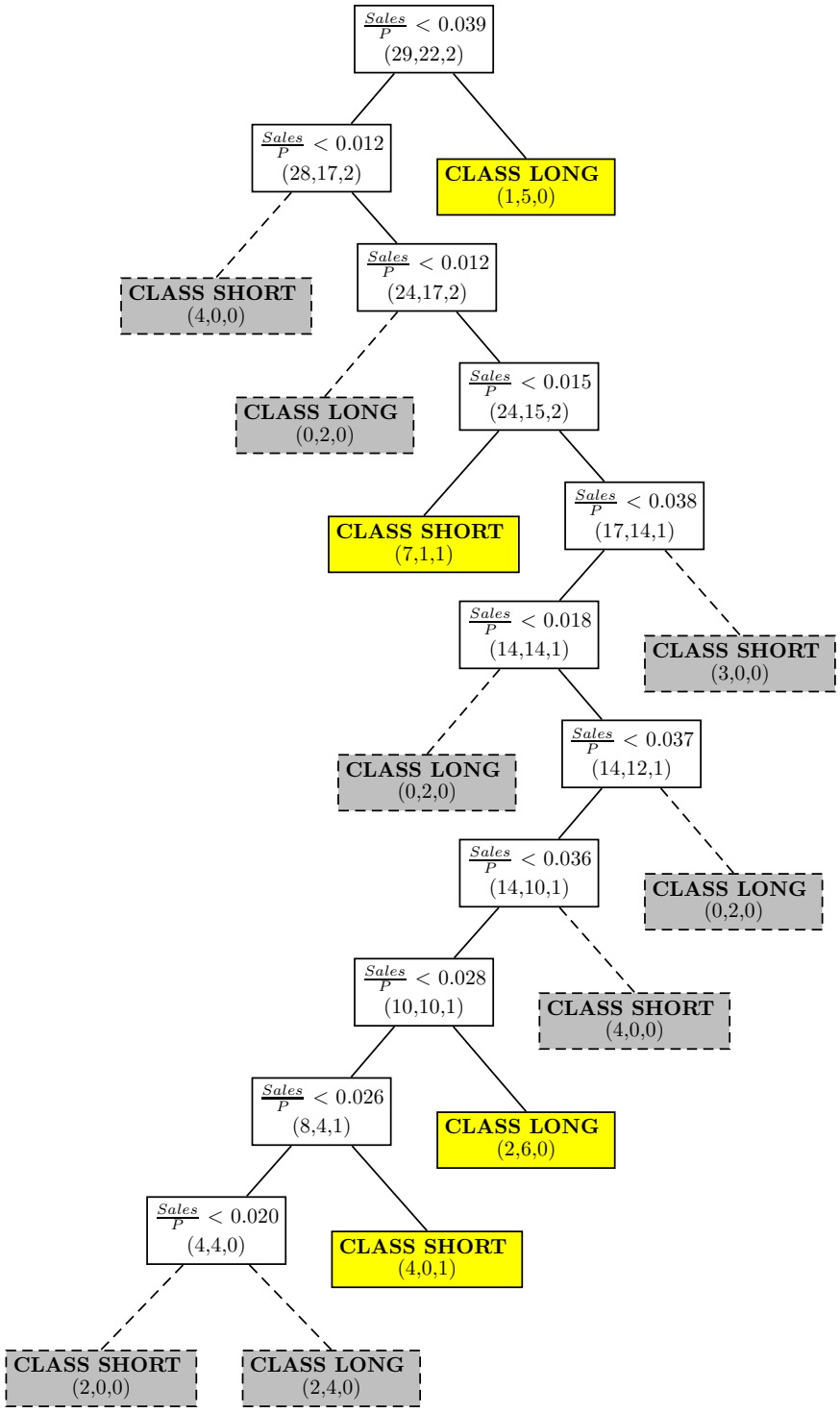

Figure 10: ALV stock classification tree, BNS pruning

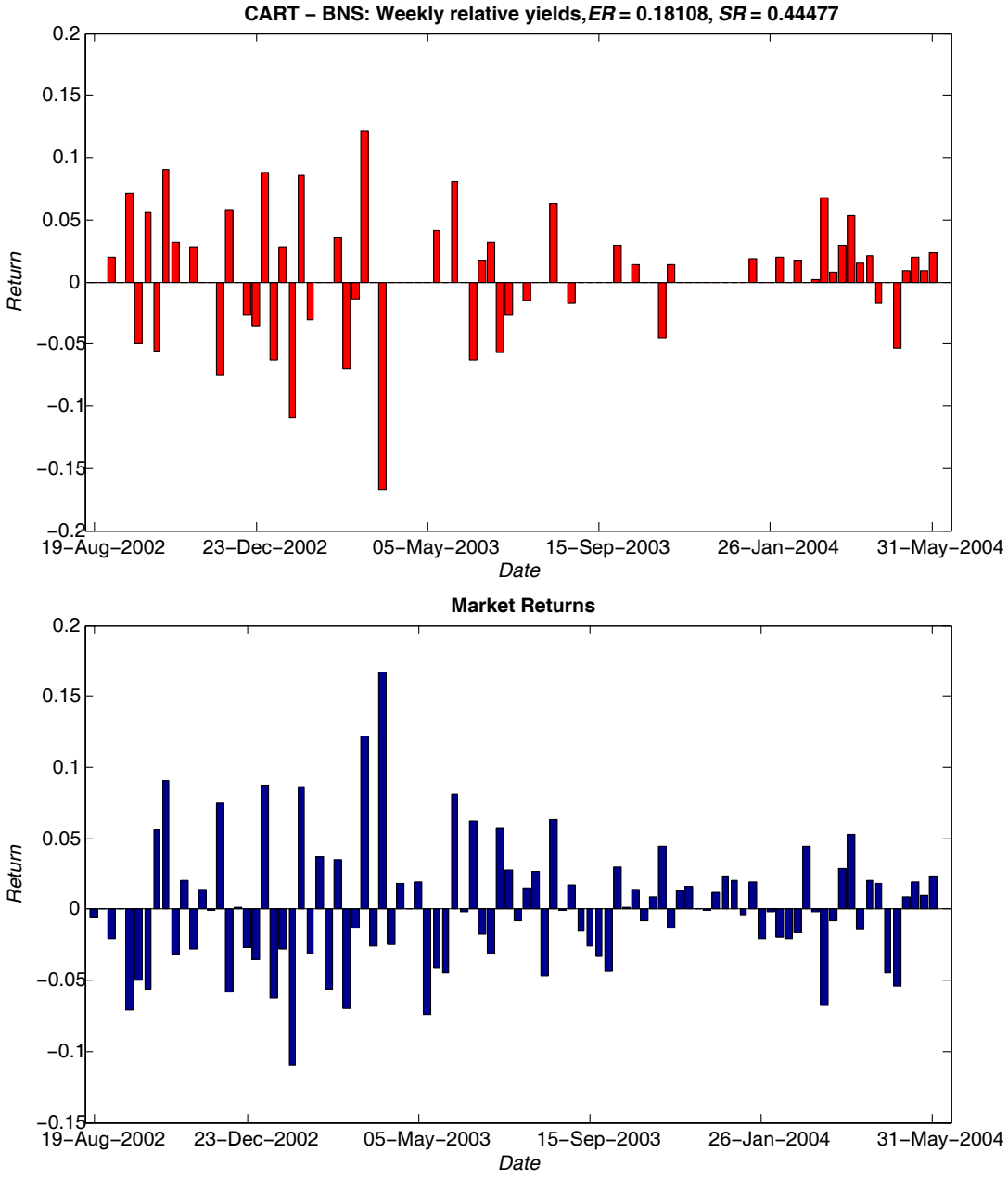

Figure 11: Backtesting performance of the BAS stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

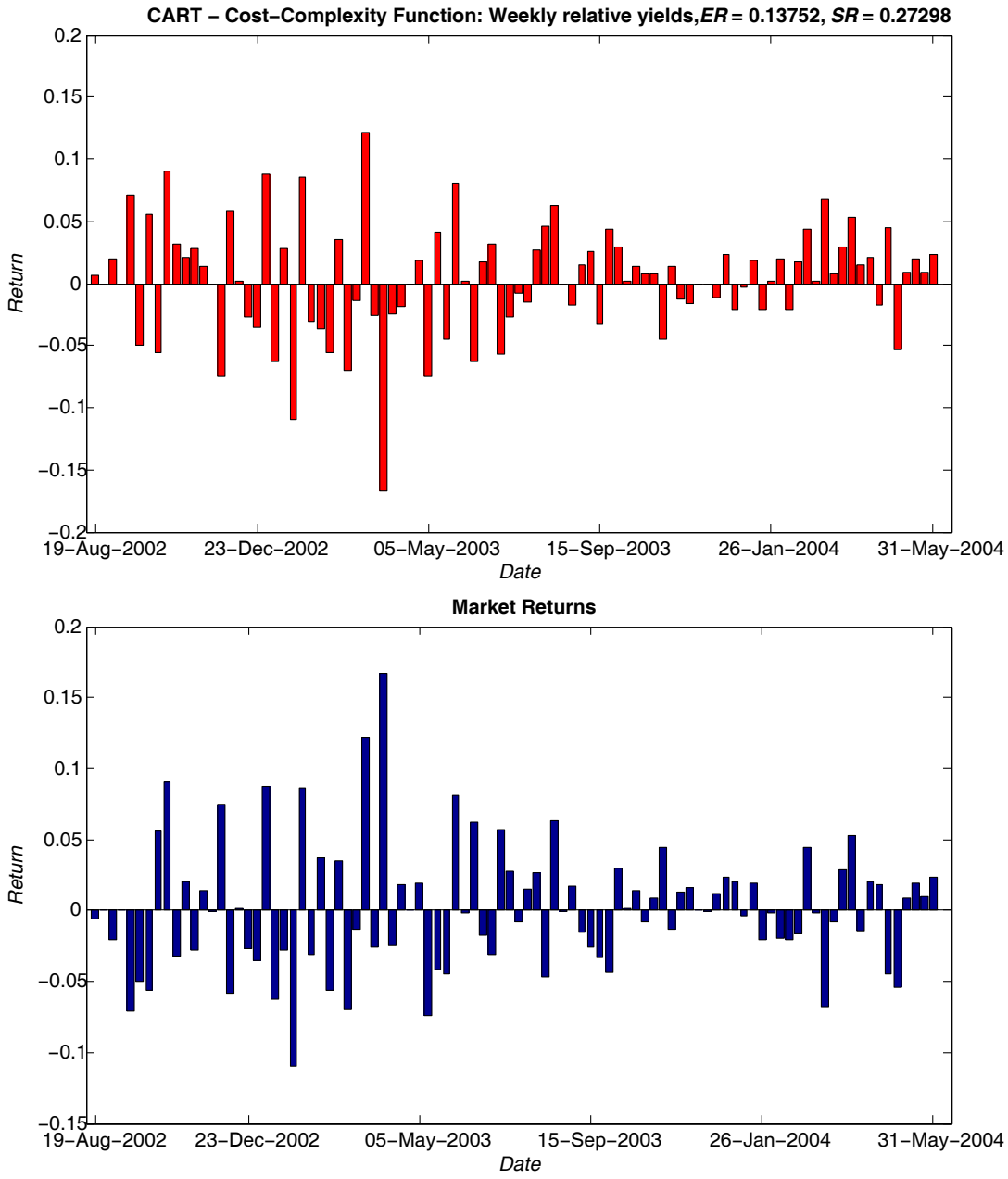

Figure 12: Backtesting performance of the BAS stock when the cost-complexity tradeoff is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

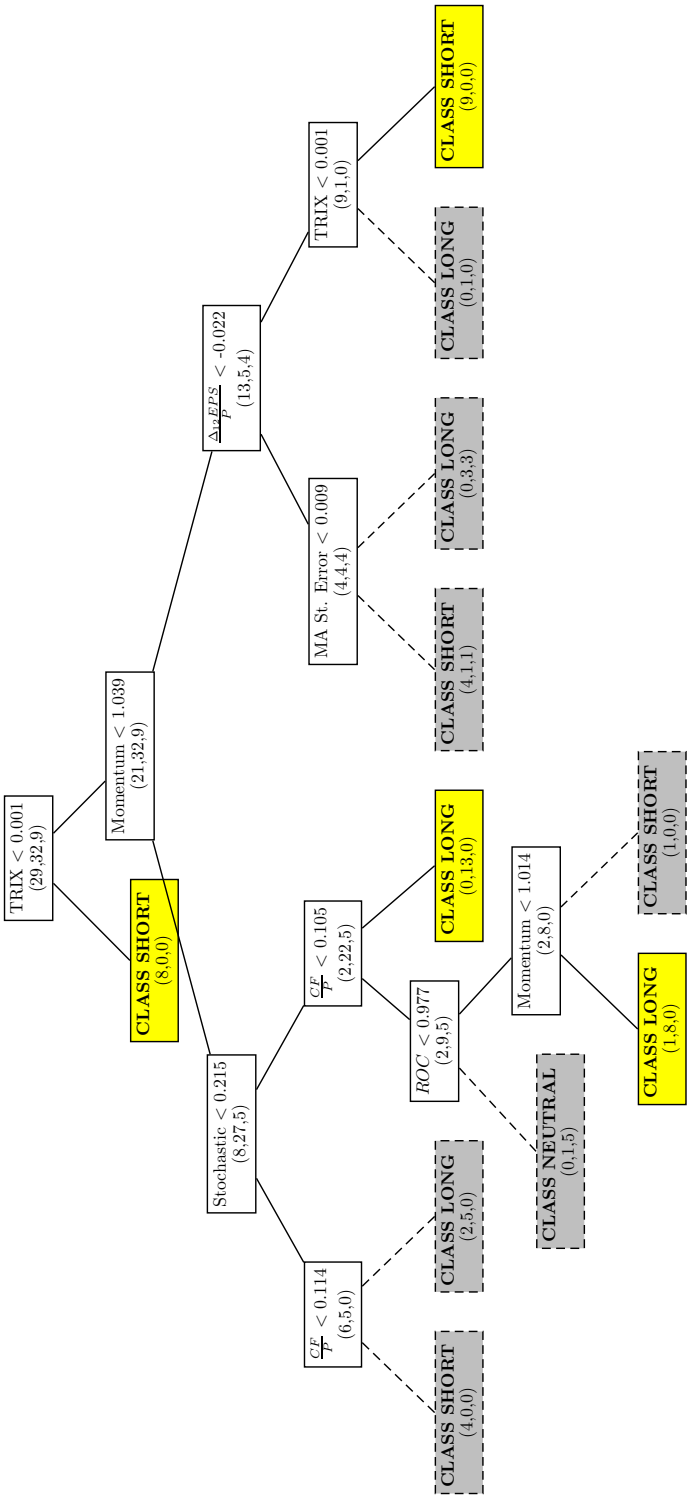

Figure 13: BAS stock classification tree, BNS pruning

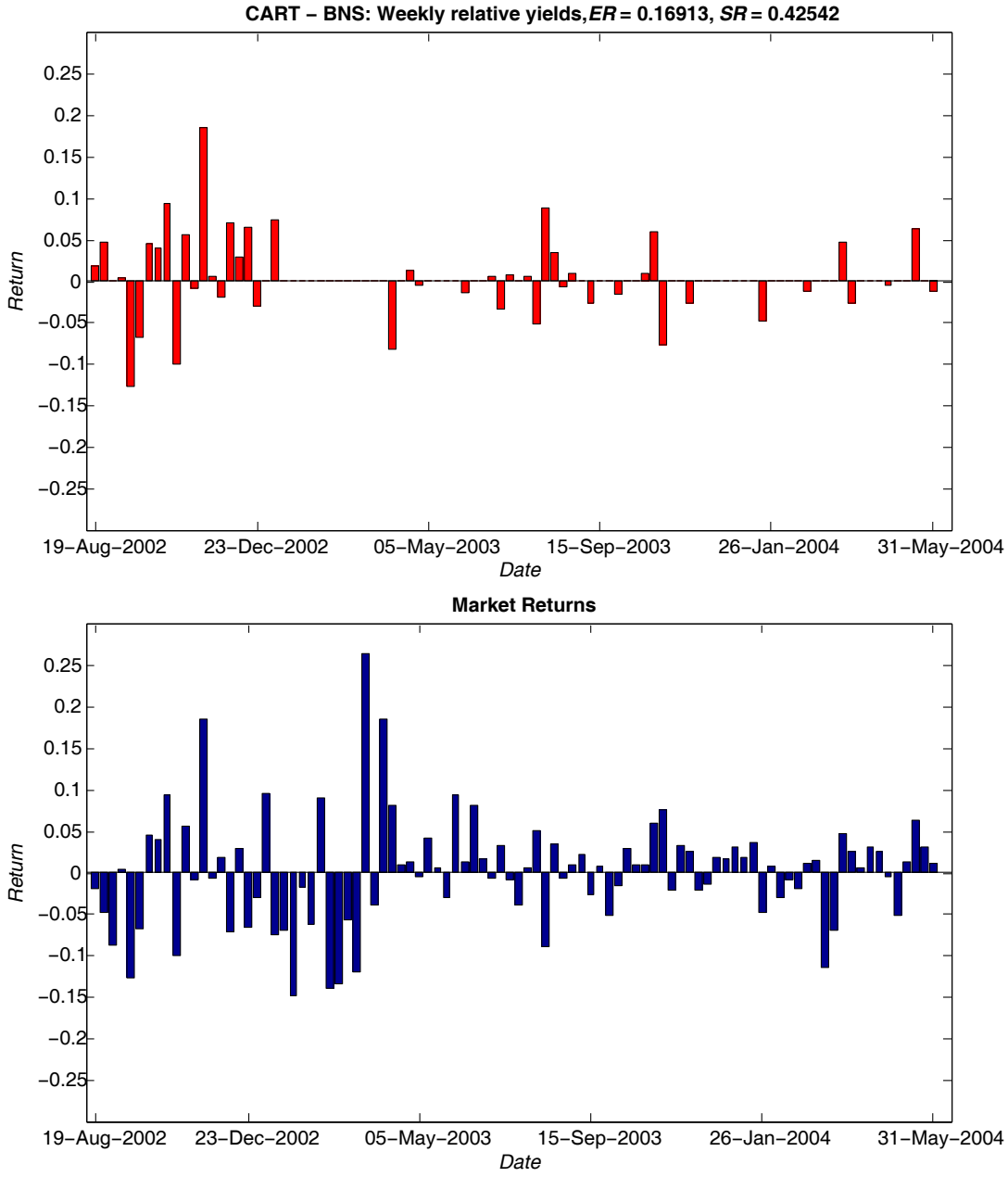

Figure 14: Backtesting performance of the BAY stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

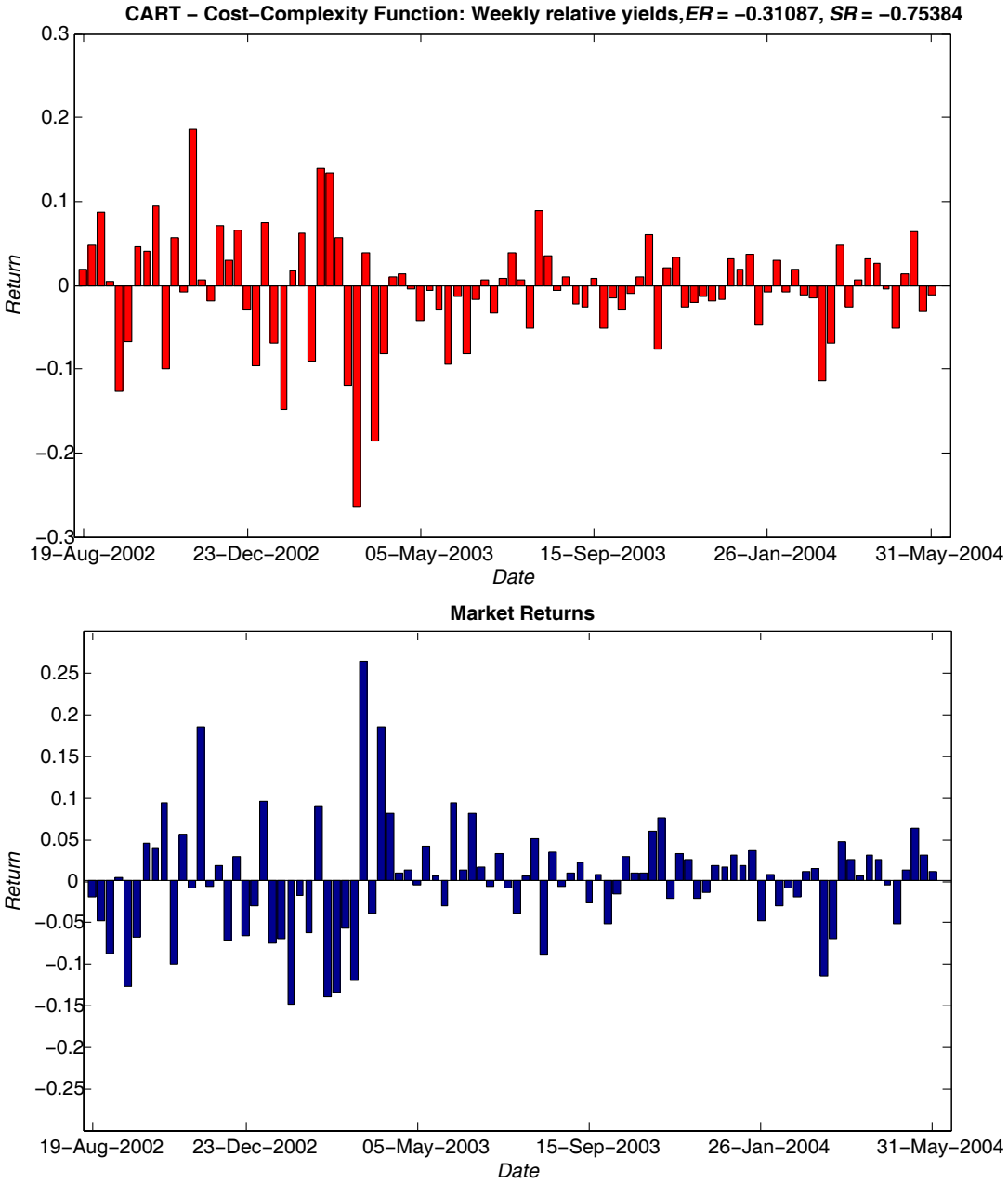

Figure 15: Backtesting performance of the BAY stock when the cost-complexity tradeoff is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

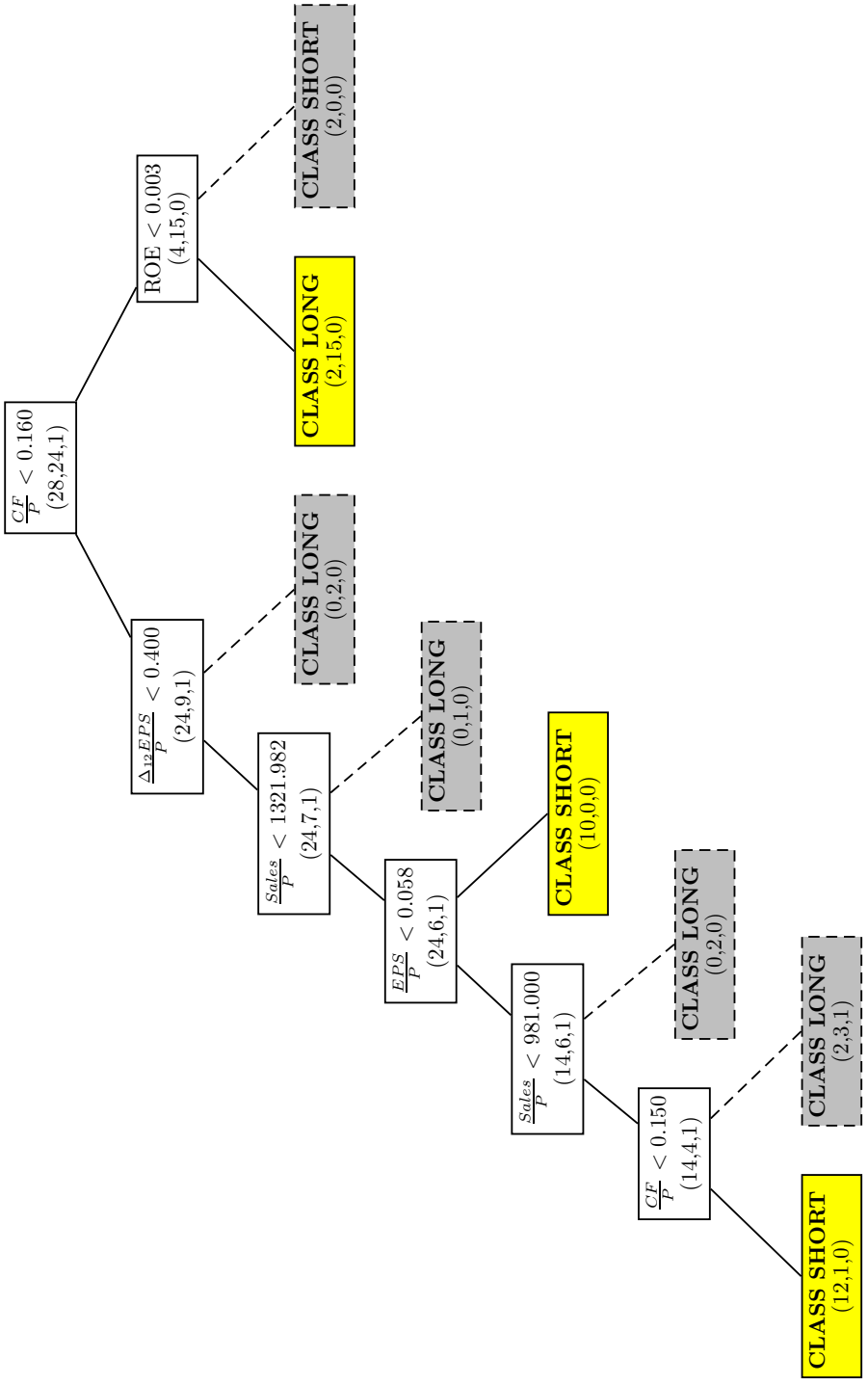

Figure 16: BAY stock classification tree, BNS pruning

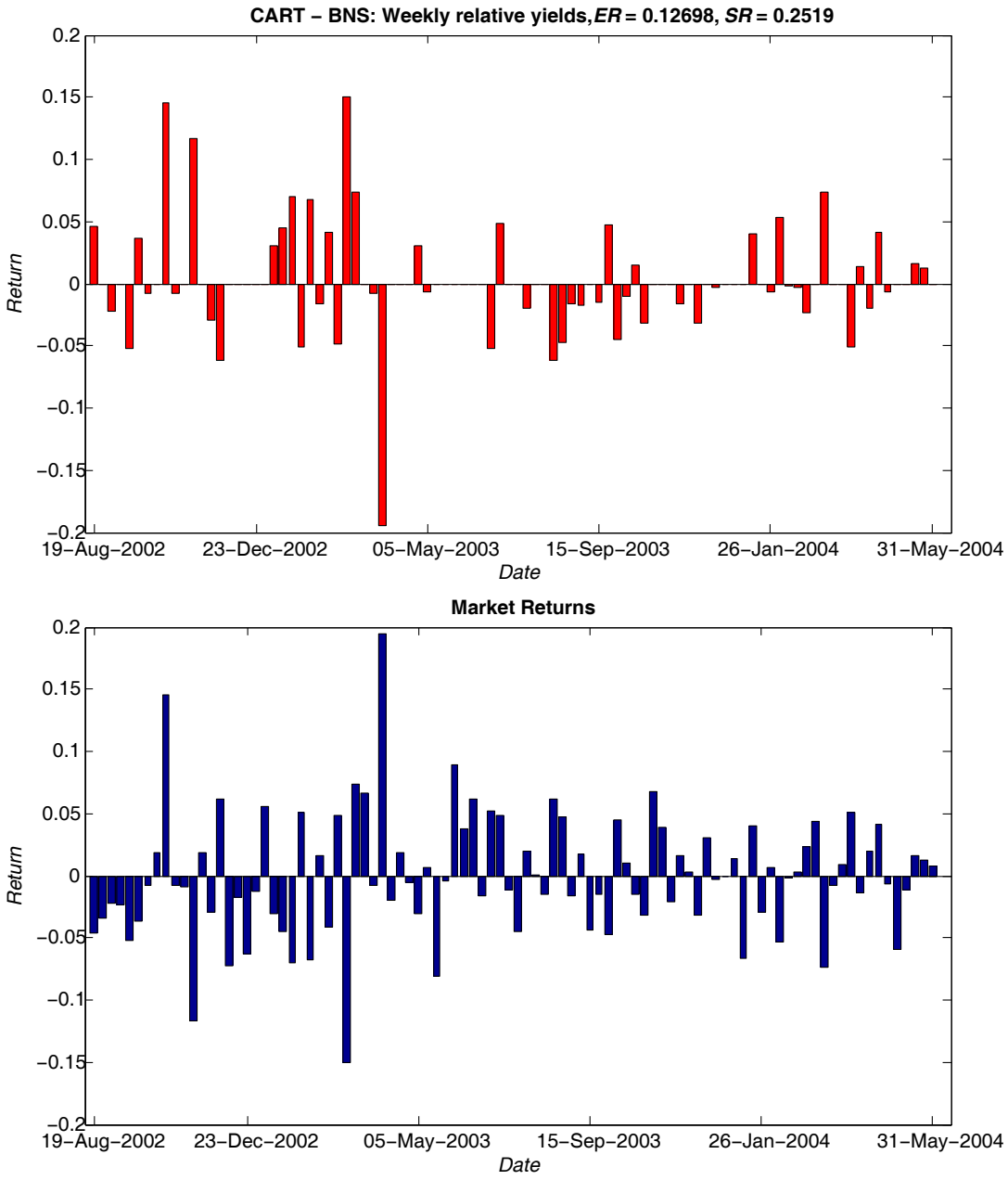

Figure 17: Backtesting performance of the BMW stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

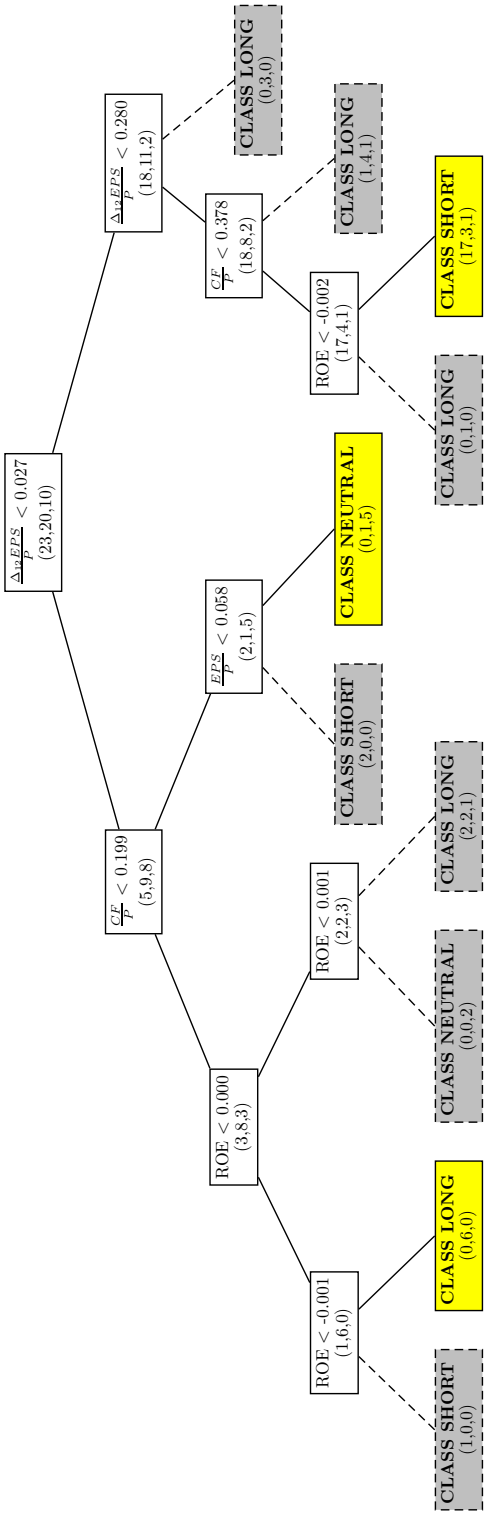

Figure 18: BMW stock classification tree, BNS pruning

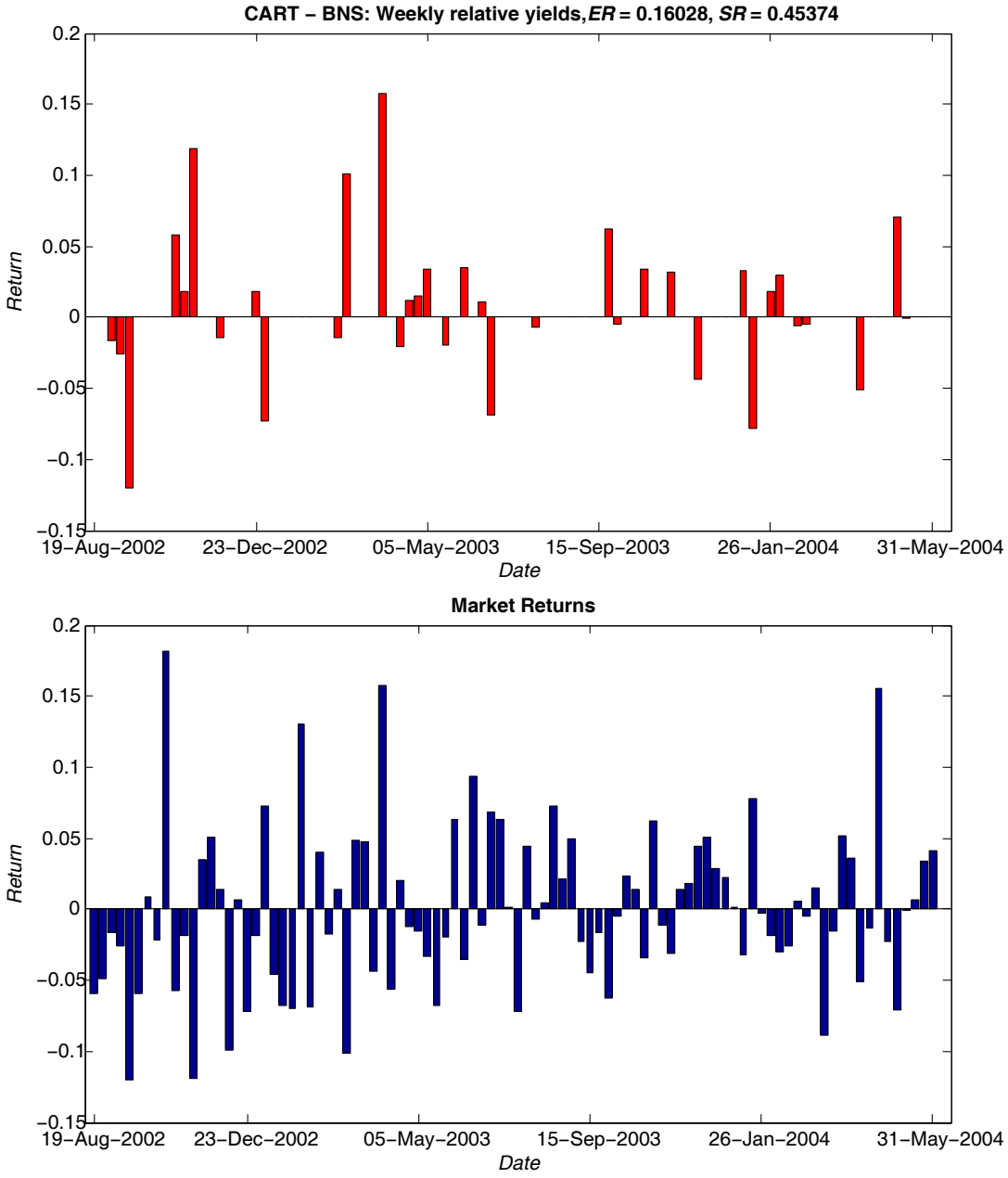

Figure 19: Backtesting performance of the DCX stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

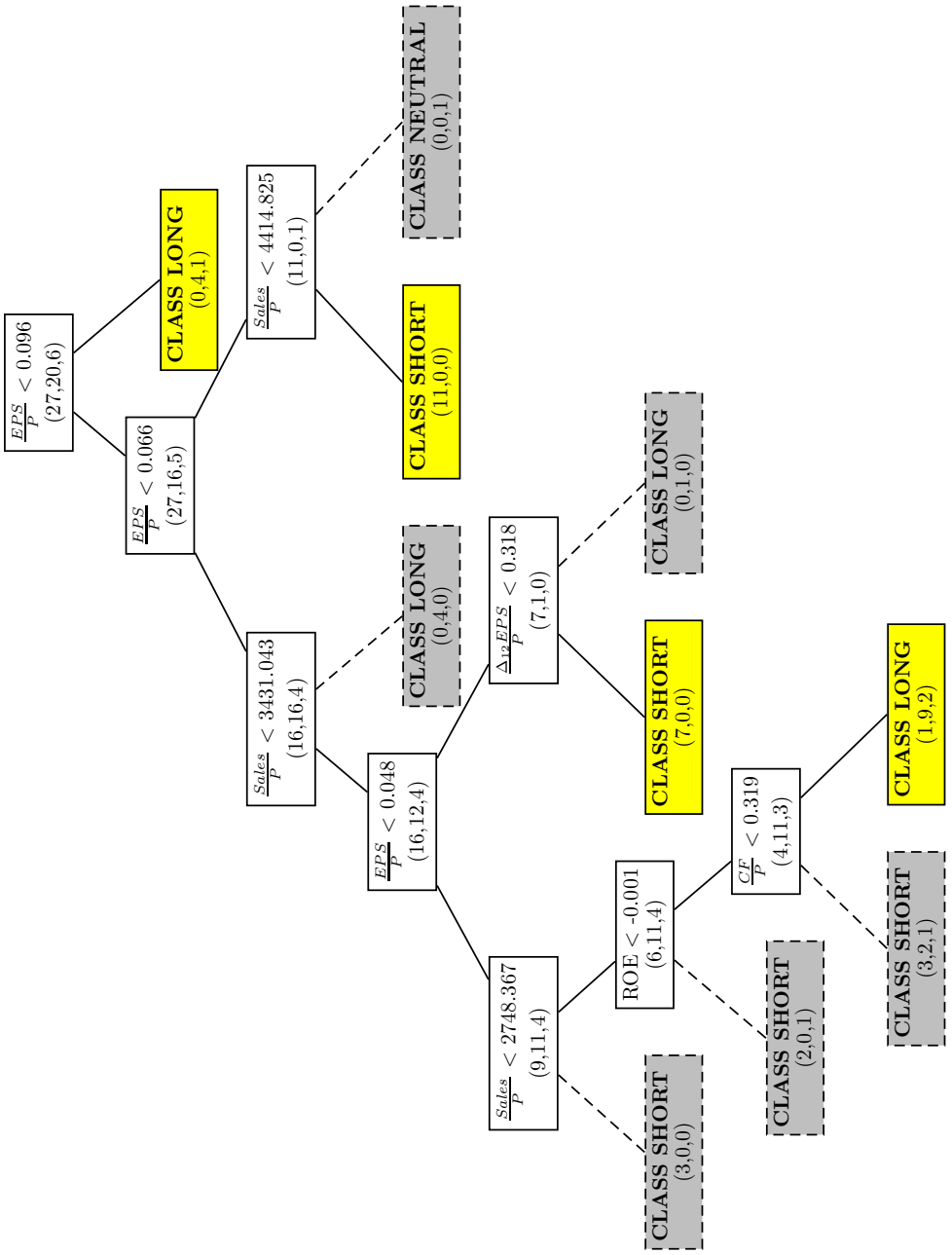

Figure 20: DCX stock classification tree, BNS pruning

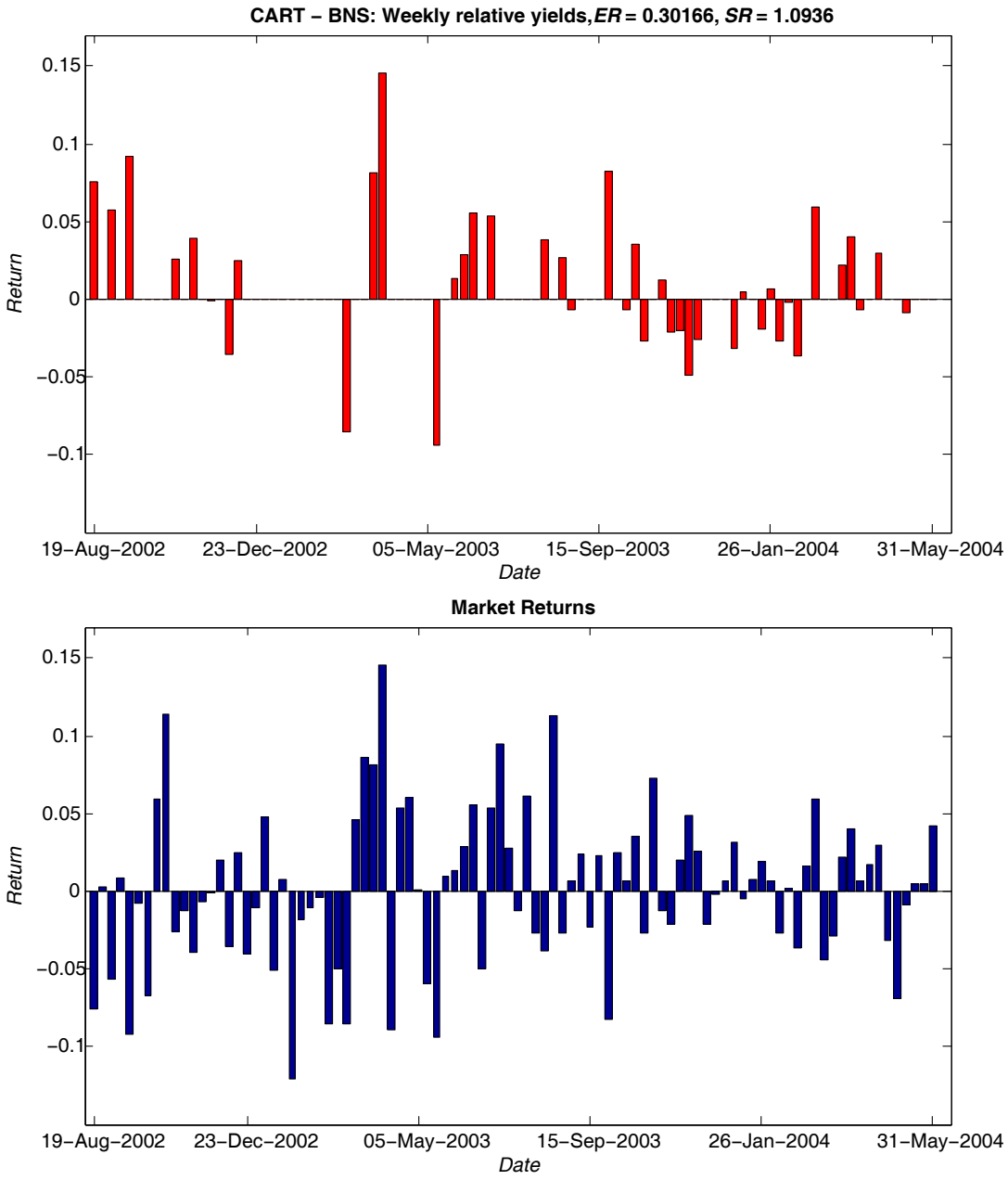

Figure 21: Backtesting performance of the LIN stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

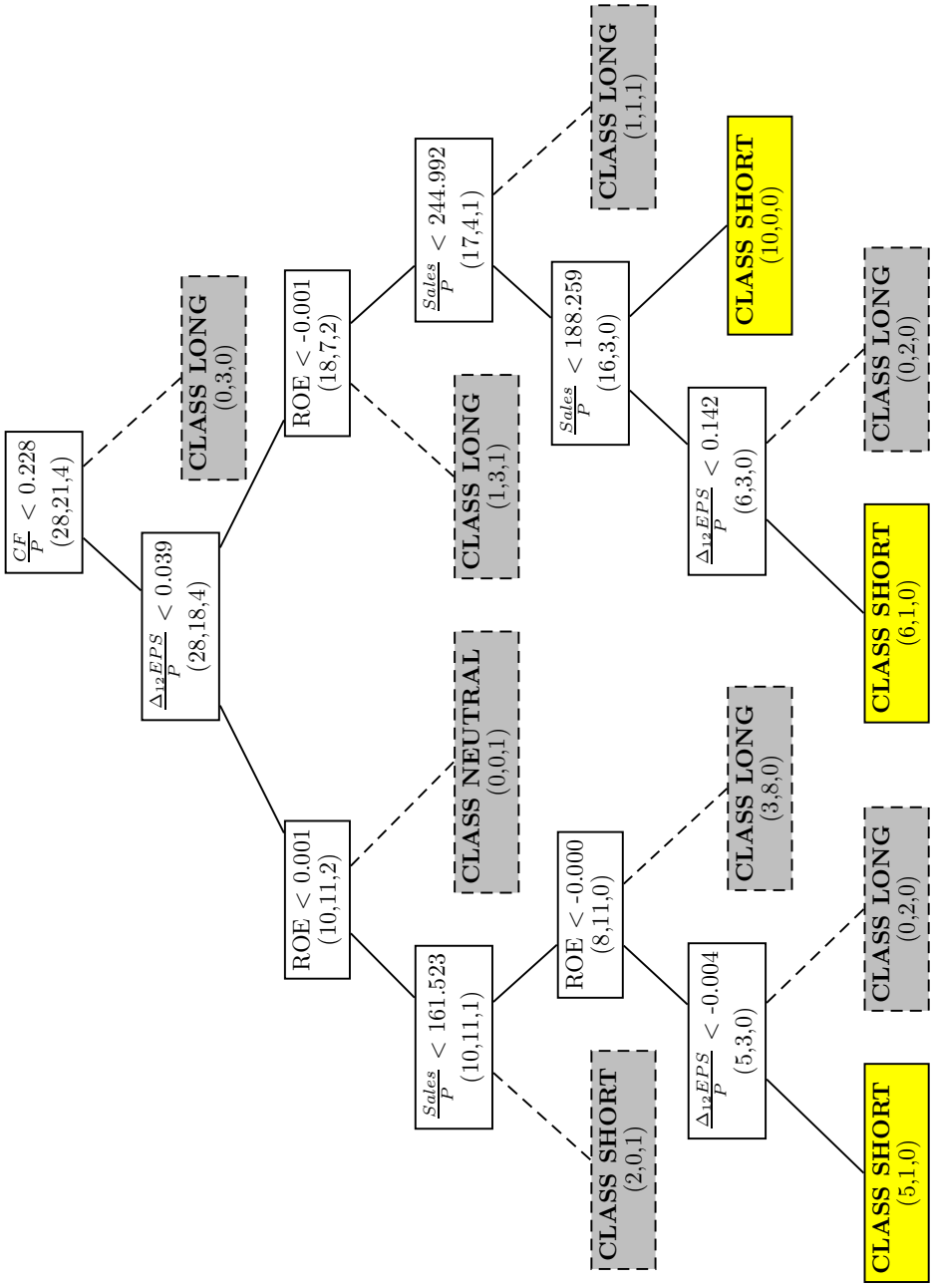

Figure 22: LIN stock classification tree, BNS pruning

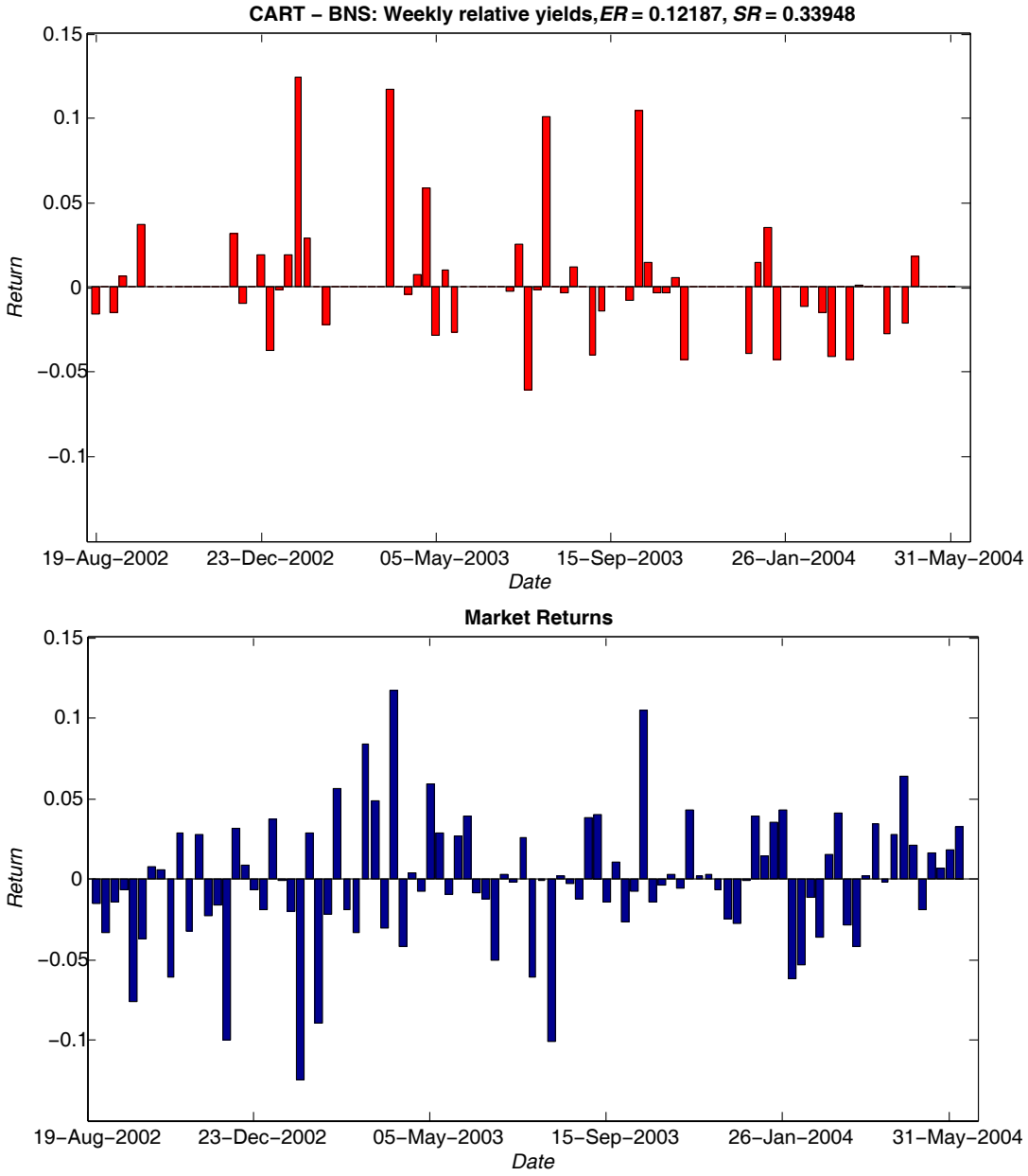

Figure 23: Backtesting performance of the SCH stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

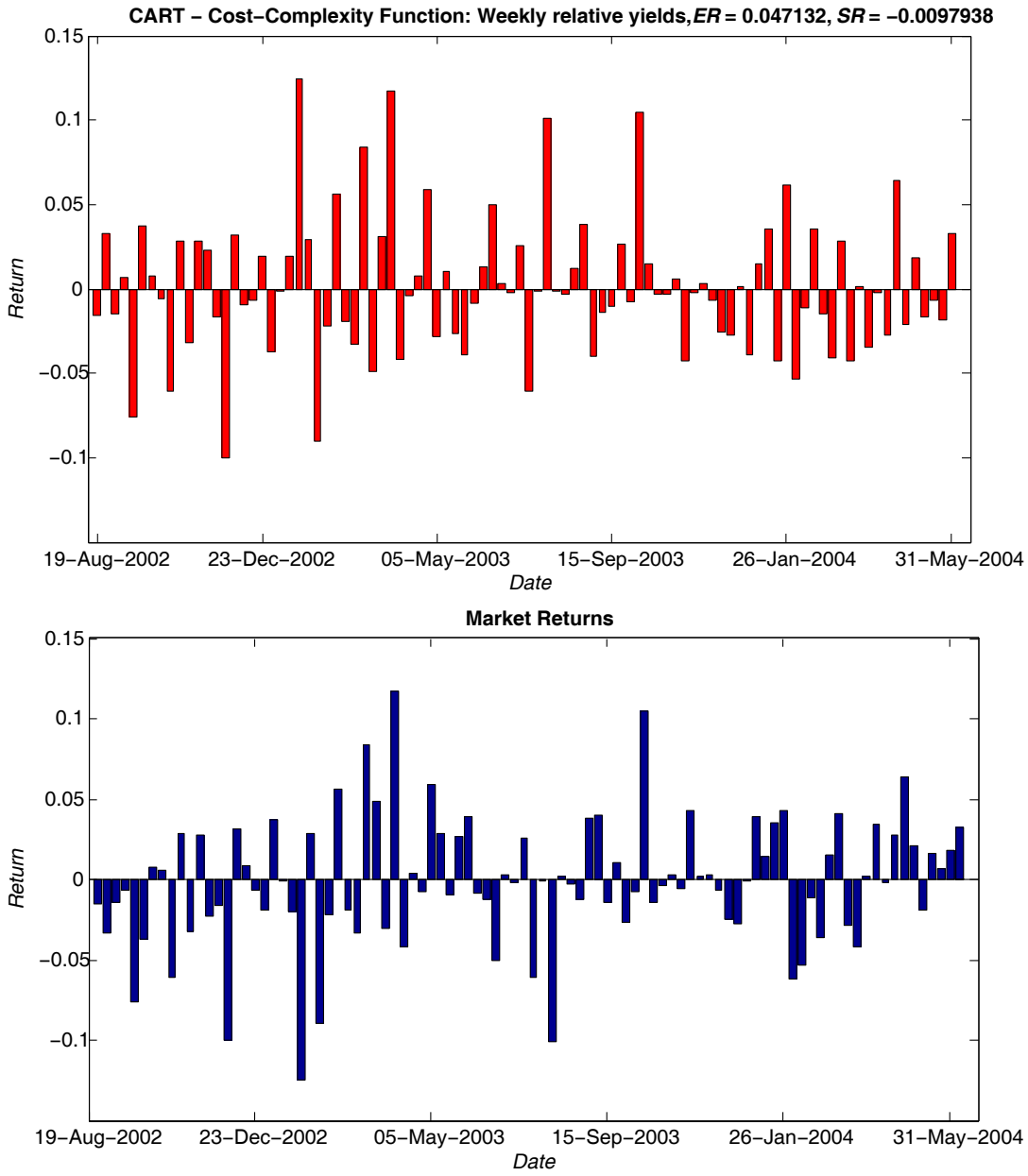

Figure 24: Backtesting performance of the SCH stock when the cost-complexity tradeoff is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

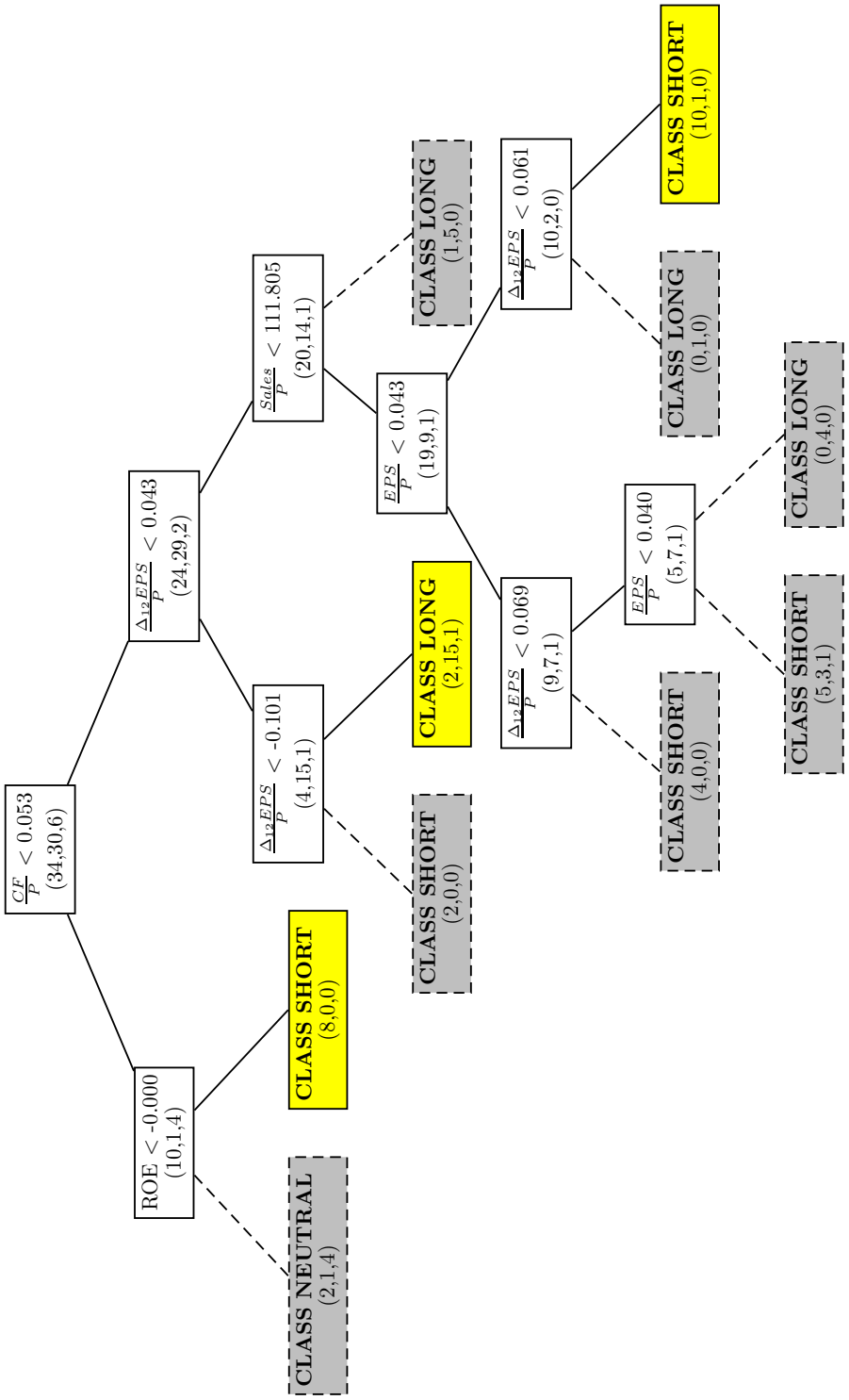

Figure 25: SCH stock classification tree, BNS pruning

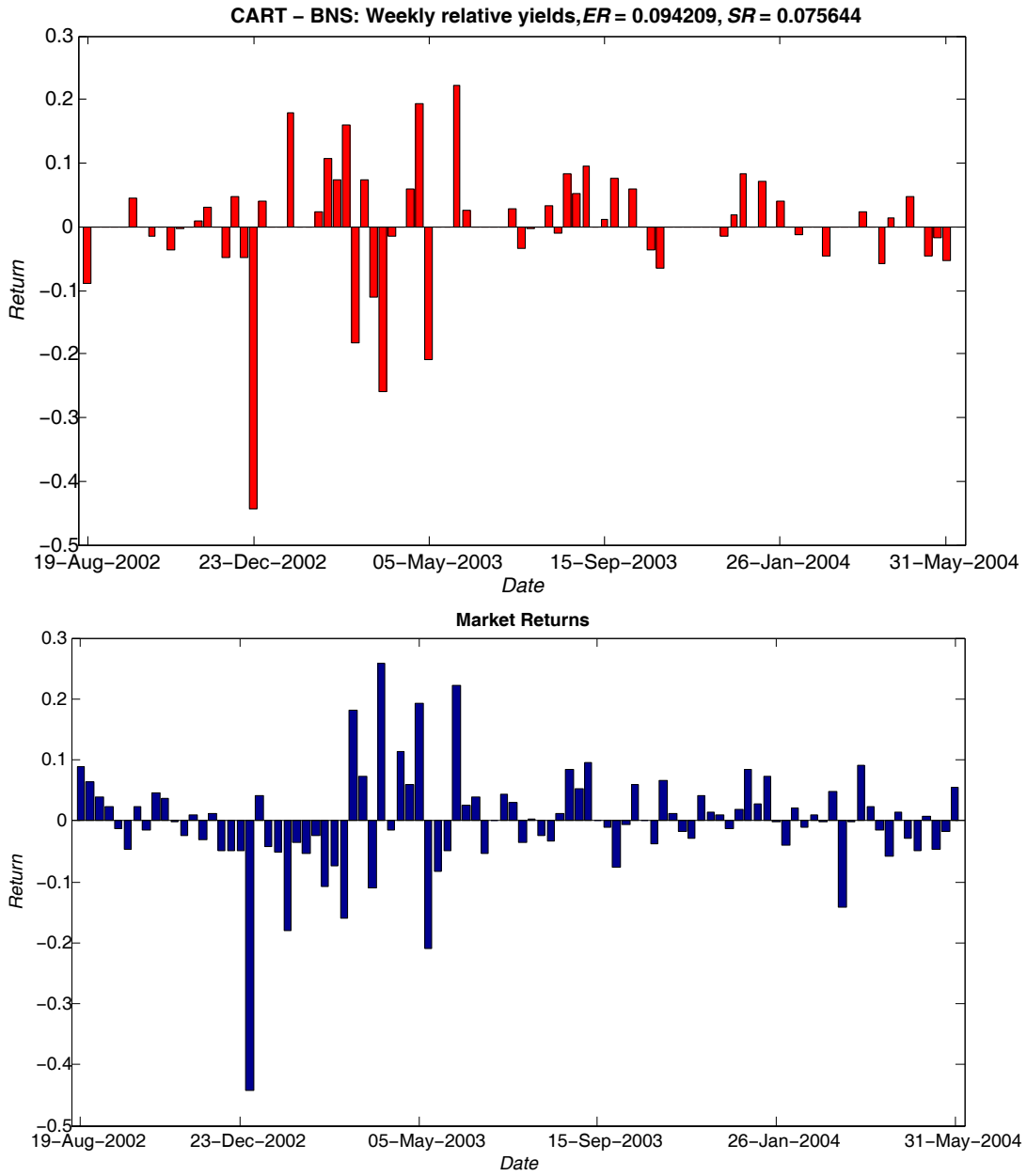

Figure 26: Backtesting performance of the TUI stock when BNS is employed for tree pruning (upper figure) and market returns of the same stock (lower figure). ER stands for the annualized expected return, SR is the Sharpe ratio

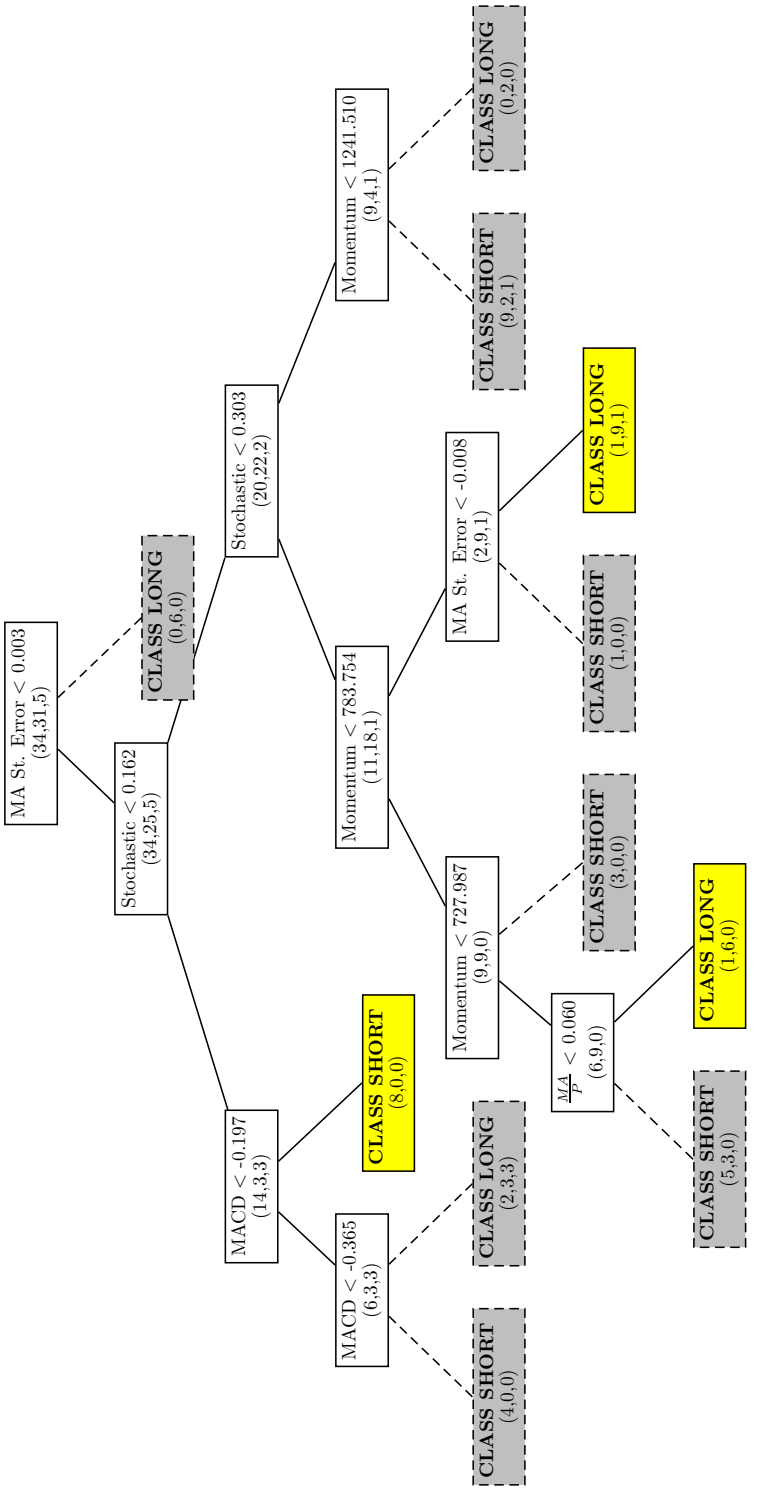

Figure 27: TUI stock classification tree, BNS pruning

Bibliography

A. Ahmadi, S. Omatu, T. Fujinaka, and T. Kosaka. Improvement of reliability in banknote classification using reject option and local PCA. *Information Sciences*, 168 (1–4):277–293, 2004.

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

N. Amenc, P. Malaise, L. Martellini, and D. Sfeir. Portable alpha and portable beta strategies in the eurozone: Implementing active asset allocation decisions using equity index options and futures. *Edhec Business School*, 2003.

D. W. K. Andrews. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica*, 59(3):817–858, 1991.

R. Balvers, T. Cosimano, and B. McDonald. Predicting stock returns in an efficient market. *The Journal of Finance*, 45(4):1109–1128, 1990.

J. Blazewicz, W. Kubiak, T. Morzy, and M. Rusinkiewicz. *Handbook on Data Management in Information Systems*. Springer-Verlag Berlin and Heidelberg GmbH & Co., 2003.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

M. Bramer. Using J-pruning to reduce overfitting in classification trees. *Knowledge-Based Systems*, 15(5–6):301–308, 2002.

L. Breiman. [Neural networks: A review from statistical perspective]: Comment. *Statistical Science*, 9(1):38–42, 1994.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a.

L. Breiman. Out-of-bag estimation. Technical report, Statistics Department, University of California, 1996b. URL `ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps`.

L. Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, 1996c.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman and J. H. Friedman. Tree-structured classification via generalized discriminant analysis: Comment. *Journal of the American Statistical Association*, 83(403): 725–727, 1988.

L. Breiman, J. H. Friedman, A. R. Olshen, and J.C. Stone. *Classification and regression trees*. The Wadsworth Statistics/Probability Series, 1987.

N. Brennan, P. Parameswaran, J. Gadaut, A. Luck, M. Dowle, S. Fagg, G. Brar, M. Turner, E. Flynn, D. Jessop, K. Yu, and A. McCutcheon. A method for selecting stocks within sectors. *Salomon Smith Barney Equity Research: Europe, Quantitative Strategy*, 1999.

W. R. Burrows, M. Benjamin, S. Beauchamp, E. R. Lord, D. McCollor, and B. Thomson. CART decision-tree statistical analysis and prediction of summer season maximum surface ozone for the Vancouver, Montreal, and Atlantic Regions of Canada. *Journal of Applied meteorology*, 34(8):1848–1862, 1995.

J. Campbell and Y. Hamao. Predictable stock returns in the United States and Japan: A study of long-term capital market integration. *The Journal of Finance*, 47(1):43–69, 1992.

N. Chen. Financial investment opportunities and the macroeconomy. *The Journal of Finance*, 46(2):529–554, 1991.

C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Knowledge Discovery and Data Mining*, pages 84–93, 1999. URL `citeseer.ist.psu.edu/cheng99entropybased.html`.

D. C. Cho, C. S. Eun, and L. W. Senbet. International arbitrage pricing theory: An empirical investigation. *The Journal of Finance*, 41(2):313–329, 1986.

G. C. Chow. Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, 28(3):591–605, 1960.

J. H. Cochrane. *Asset pricing*. Princeton University Press, 2005.

J. Conrad and G. Kaul. An anatomy of trading strategies. *The Review of Financial Studies*, 11(3):489–519, 1998.

R. Davidson and J. G. MacKinnon. *Econometric Theory and Methods*. Oxford University Press, 2004.

F. X. Diebold and R. S. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13(3):253–265, 1995.

T. G. Dietterich. Ensemble methods in machine learning. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, London, UK, 2000a. Springer-Verlag.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000b.

G. Dutta, P. Jha, A. K. Laha, and N. Mohan. Artificial neural network models for forecasting stock price index in the Bombay stock exchange. *Journal of Emerging Market Finance*, 5(3):283–295, 2006.

F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

E. Fama. Efficient capital markets: II. *The Journal of Finance*, 46(5):1575–1617, 1991.

E. Fama and K. French. Dividend yields and expected stock returns. *Journal of Financial Economics*, 22(1):3–25, 1988a.

E. Fama and K. French. Permanent and temporary components of stock prices. *Journal of Political Economy*, 96(2):246–73, 1988b.

E. Fama and K. French. The cross-section of expected stock returns. *The Journal of Finance*, 47(2):427–465, 1992.

E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970.

E. F. Fama. Multifactor portfolio efficiency and multifactor asset pricing. *Journal of Financial and Quantitative Analysis*, 31(4):441–465, 1996.

E. F. Fama. Determining the number of priced state variables in the icapm. *The Journal of Financial and Quantitative Analysis*, 33(2):217–231, 1998a.

E. F. Fama. Market efficiency, long-term returns, and behavioral finance. *Journal of Financial Economics*, 49(3):283–306, 1998b.

E. F. Fama and K. R. French. The CAPM is wanted, dead or alive. *The Journal of Finance*, 51(5):1947–1958, 1996.

E. F. Fama and K. R. French. The equity premium. *The Journal of Finance*, 57(2): 637–659, 2002.

S. Feldman, D. F. Klein, and G. Honigfeld. The reliability of a decision tree technique applied to psychiatric diagnosis. *Biometrics*, 28(3):831–840, 1972.

W. Ferson and C. Harvey. The variation in economic risk premia. *Journal of Political Economy*, 99:385–415, 1991.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

E. Fix and J. Hodges. Discriminatory analysis: Nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

W. H. Greene. *Econometric Analysis*. Prentice Hall, 1997.

P. D. Grünwald and J. Rissanen. *The Minimum Description Length Principle*. MIT Press, 2007.

B. Hanczar and E. R. Dougherty. Classification with reject option in gene expression data. *Bioinformatics*, 24(17):1889–1895, 2008.

W. Härdle. *Applied Nonparametric Regression*. Cambridge University Press, 1990.

W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer, 2003.

W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer, 2004.

W. Härdle, Y. Mori, and P. Vieu. *Statistical Methods for Biostatistics and Related Fields*. Springer, 2007.

G. Harman, P. Parameswaran, and M. Witt. Shares, bonds or cash? Asset allocation in the new economy using CART. *Salomon Smith Barney Equity Research: Australia, Quantitative Analysis*, 2000.

M. L. Hartzmark. Luck versus forecast ability: Determinants of trader performance in futures markets. *The Journal of Business*, 64(1):49–74, 1991.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

J. A. Hausman. Specification tests in econometrics. *Econometrica*, 46(6):1251–1271, 1978.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

R. J. Hodrick. Dividend yields and expected stock returns: Alternative procedures for inference and measurement. *The Review of Financial Studies*, 5(3):357–386, 1992.

Y. Hong and H. White. Consistent specification testing via nonparametric series regression. *Econometrica*, 63(5):1133–1159, 1995.

Z. Huang, H. Chen, C.-J. Hsu, W.-H. Chen, and S. Wu. Credit rating analysis with Support Vector Machines and Neural Networks: A market comparative study. *Decision Support Systems*, 37(4):543–558, 2004.

A. Ittner and M. Schlosser. Non-linear decision trees – NDT. In *ICML*, pages 252–257, 1996a.

Andreas Ittner and Michael Schlosser. Non-linear decision trees – NDT. In *International Conference on Machine Learning*, pages 252–257, 1996b. URL `citeseer.ist.psu.edu/412983.html`.

N. Jegadeesh. Evidence of predictable behavior of security returns. *The Journal of Finance*, 45(3):881–898, 1990.

D. Johnson and R. McClelland. Nonparametric tests for the independence of regressors and disturbances as specification tests. *The Review of Economics and Statistics*, 79 (2):335–340, 1997.

G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

D. Keim and R. Stambaugh. Predicting returns in the stock and bond markets. *Journal of Financial Economics*, 17:357–390, 1986.

H. Kim and W.-Y. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96(454):589–604, 2001.

S. Kim and S. S. Oh. Decision-tree-based Markov model for phrase break prediction. *ETRI Journal*, 29(4):527–529, 2007.

I. Kolyshkina and R. Brookes. Data mining approaches to modelling insurance risk. *PricewaterhouseCoopers*, 2002.

J.R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 2007.

H. Levene. Robust tests for equality of variances. In I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, editors, *Contributions to Probability and Statistics*, pages 278–292. Stanford University Press, Palo Alto, 1960.

J. Lewellen and J. Shanken. Learning, asset-pricing tests, and market efficiency. *The Journal of Finance*, 57(3):1113–1145, 2002.

K. K. Lewis. Changing beliefs and systematic rational forecast errors with evidence from foreign exchange. *The American Economic Review*, 79(4):621–636, 1989.

T.-S. Lim and W.-Y. Loh. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–229, 2000.

J. Lintner. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *The Review of Economics and Statistics*, 47(1):13–37, 1965a.

J. Lintner. Security prices, risk, and maximal gains from diversification. *The Journal of Finance*, 20(4):587–615, 1965b.

W.-Y. Loh and Y.-S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.

W.-Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83(403):715–725, 1988a.

W.-Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis: Rejoinder. *Journal of the American Statistical Association*, 83(403): 728, 1988b.

H. M. Markowitz. *Portfolio selection: efficient diversification of investment.* Blackwell Publishers Ltd, Oxford, UK, 1991.

M. Mehta, J. Rissanen, and R. Agrawal. MDL-based decision tree pruning. In *In Int'l Conference on Knowledge Discovery in Databases and Data Mining (KDD-95)*, pages 216–221. AAAI Press, 1995.

R. C. Merton. An intertemporal capital asset pricing model. *Econometrica*, 41(5): 867–887, 1973.

J. Mingers. Expert systems – rule induction with statistical data. *Journal of the Operational Research Society*, 38:39–47, 1987.

J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989a.

J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989b.

J. N. Morgan and R. C. Messenger. *THAID – a sequential analysis program for the analysis of nominal scale dependent variables.* Survey Research Center, Institute for Social Research, University of Michigan, 1973.

J. N. Morgan and J. A. Sonquist. Some results from a non-symmetrical branching process that looks for interaction effects. In *Proceedings of the Social Statistics Section of the American Statistical Association*, pages 40–53, 1963a.

J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434, 1963b.

S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

S. N. Neftci. Naive trading rules in financial markets and Wiener-Kolmogorov prediction theory: A study of 'technical analysis'. *The Journal of Business*, 64(4):549–571, 1991.

W. K. Newey and K. D. West. Automatic lag selection in covariance matrix estimation. *The Review of Economic Studies*, 61(4):631–653, 1994.

T. Niblett and I. Bratko. Learning decision rules in noisy domains. In *Proceedings of Expert Systems '86, The 6th Annual Technical Conference on Research and Development in Expert Systems III*, pages 25–34, New York, NY, USA, 1987. Cambridge University Press.

K.-M. Osei-Bryson. Evaluation of decision trees: a multi-criteria approach. *Computers & Operations Research*, 31(11):1933–1945, 2004.

K.-M. Osei-Bryson. Post-pruning in decision tree induction using multiple performance measures. *Computers & Operations Research*, 34(11):3331–3345, 2007.

E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

E. A. Patrick. *Fundamentals of Pattern Recognition.* Prentice-Hall, 1972.

A. A. Petrov, I. G. Pospelov, and A. A. Shananin. *The experience of mathematical modeling of the economy.* Energoatomizdat, Moscow, Russia, 1996.

L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The mathematical theory of optimal processes.* New York: Interscience Publishers, John Wiley & Sons, 1962.

I. G. Pospelov. *Modeling of economic structures.* Fazis * CC RAS, Moscow, Russia, 2003.

J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

J. R. Quinlan. Bagging, boosting, and C4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

P. M. Robinson. Nonparametric methods in specification. *The Economic Journal*, 96 (134–141), 1986.

B. P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2–3):577–584, 2005.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory, Psychological Review*, 65 (6):386–408, 1958.

S. A. Ross. The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3):341–360, 1976.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Quantitative Equity Products Team at Schroders. Tree building at Schroders. *Schroder Investment Management Limited: Quantitative Equity Products*, 2006.

L. Seshadri. JPMorgan US quantitative factor model: August 2003 stock list. *JPMorgan Quantitative Equity and Derivatives*, 2003.

W. F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442, 1964.

W. F. Sharpe. The Sharpe ratio. *Journal of Portfolio Management*, 21(1):49–58, 1994.

R. J. Shiller. *Market Volatility.* MIT Press, 1989.

R. J. Shiller. Market volatility and investor behavior. *The American Economic Review*, 80(2):58–62, 1990.

D. A. Simovici, D. Cristofor, and L. Cristofor. Impurity measures and applications to classification and clustering, 2000. URL `citeseer.ist.psu.edu/460048.html`.

H. E. Sorensen, K. C. Ooi, and L. K. Miller. The decision tree approach to stock selection. *Salomon Smith Barney Equity Research: United States, Global Quantitative Research*, 1999.

H. J. Steadman, E. Silver, J. Monahan, P. S. Appelbaum, P. C. Robbins, E. P. Mulvey, T. Grisso, L. H. Roth, and S. Banks. A classification tree approach to the development of actuarial violence risk assessment tools. *Law and Human Behavior*, 24(1):83–100, 2000.

M. C. Steinbach. Markowitz revisited: Mean-variance models in financial portfolio analysis. *SIAM Review*, 43(1):31–85, 2001.

R. M. Stulz. An equilibrium model of exchange rate determination and asset pricing with nontraded goods and imperfect information. *The Journal of Political Economy*, 95(5):1024–1040, 1987.

R. Sullivan, A. Timmermann, and H. White. Data-snooping, technical trading rule performance, and the bootstrap. *The Journal of Finance*, 54(5):1647–1691, 1999.

K. Y. Tam and M. Y. Kiang. Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, 38(7):926–947, 1992.

A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems*. W. H. Winston, Washington, DC, 1977.

P. E. Utgoff. Perceptron trees: A case study in hybrid concept representations. *Connection Science*, 1(4):377–391, 1989.

P. E. Utgoff and C. E. Brodley. *Linear machine decision trees*. Tech. rep. 10, University of Massachusetts at Amherst, 1991.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

Y. Wu and Y. Liu. Robust truncated hinge loss Support Vector Machines. *Journal of the American Statistical Association*, 102(10):974–983, 2007.

Notation

Element	Description
BNS	Best Node Selection
b.p.	basis points
CART	Classification and Regression Trees
DGP	data-generating process
ER	expected return
GLS	Generalized Least Squares
LS	learning sample
LDA	Linear Discriminant Analysis
k-NN	k-Nearest Neighbor
MM	Method of Moments
OLS	Ordinary Least Squares
QDA	Quadratic Discriminant Analysis
SR	Sharpe ratio
SVM	Support Vector Machine
$e(t)$	internal node t misclassification error
$E(T)$	internal tree T misclassification error
$\mathbb{E}(X)$	expected value of X
h	bandwidth
$i(\cdot)$	impurity function
$\Delta i(\cdot)$	increment of the value of the impurity function
$\mathbf{I}(\cdot)$	indicator function
$\boldsymbol{I}$	identity matrix
J	number of classes in the learning sample
$K(\cdot)$	kernel function
n	number of observations in the (learning) sample
$n(t)$	number of observations in node t
$n_j(t)$	number of observations of class j in node t
n_{LS}	size of the learning sample

Element	Description
$\bar{n}$	BNS target node representativity level
$\mathbf{0}$	zero matrix
$\mathcal{O}$	$\alpha_n = \mathcal{O}(\beta_n)$ means: $\lim_{n\to\infty} \frac{\alpha_n}{\beta_n} = c$, c – constant
P_t	stock price at time period t
$\bar{p}$	BNS target node purity level
p_L	probability of getting to the left child node
p_R	probability of getting to the right child node
$p\left(j\|t\right)$	proportion of observations of class j in node t
Π	portfolio return
R_t	one period forward-looking price yield at time t
$\bar{R}$	threshold stock price yield
R_f	risk-free rate
$\mathbb{R}^p$	p-dimensional Euclidian space
s	arbitrary split
s^*	optimal split for a given node
t_P	parent node in the triplet of nodes $\{t_P, t_L, t_R\}$
t_L	left child node
t_R	right child node
$t[x]$	terminal node corresponding to observation x
T	arbitrary binary decision tree
T_{MAX}	maximum tree
T^*	decision tree of optimal size
$T(n)$	decision tree having at most n observations in each terminal node
$\tilde{T}$	set of terminal nodes of an arbitrary tree T
$\left\|\tilde{T}\right\|$	number of terminal nodes of T
$\mathbb{V}(X)$	variance of X: $\mathbb{V}(X) = \mathbb{E}\left[X - \mathbb{E}(X)\right]^2$
X	explanatory variable (feature)
$\boldsymbol{X}$	matrix of explanatory variables
x	element of $\boldsymbol{X}$
Y	dependent variable (vector of classes)
y	element of Y

Selbständigkeitserklärung

Ich bezeuge durch meine Unterschrift, dass meine Angaben über die bei der Abfassung benutzten Hilfsmittel, über die mir zuteil gewordene Hilfe sowie über frühere Begutachtungen meiner Dissertation in jeder Hinsicht der Wahrheit entsprechen.